STORAGE CAPACITY OF THE LINEAR ASSOCIATOR: BEGINNINGS OF A THEORY OF COMPUTATIONAL MEMORY

Technical Report AIP - 37

Dean C. Mumme

Learning Research and Development Center
and Psychology Department
University of Pittsburgh
Pittsburgh, PA. 15260

The Artificial Intelligence and Psychology Project

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

2

STORAGE CAPACITY OF THE LINEAR ASSOCIATOR: BEGINNINGS OF A THEORY OF COMPUTATIONAL MEMORY

Technical Report AIP - 37

Dean C. Mumme

Learning Research and Development Center
and Psychology Department
University of Pittsburgh
Pittsburgh, PA. 15260

April 1987

This research was supported by the Computer Sciences Division, Office of Naval Research and DARPA under Contract Number N00014-86-K-0678; Army Research Institute, under Contract No. MD903-86-C-0149; and Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research under Contract N-0014986-K-0107. Work submitted as Ph.D. thesis to the University of Illinois.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) AIP - 37	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) Department of Psychology Pittsburgh, Pennsylvania 15213	7b ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, Virginia 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Same as Monitoring Organization 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS p4000ub201/7-4-86 PROGRAM ELEMENT NO: N/A; PROJECT NO: N/A; TASK NO: N/A; WORK UNIT ACCESSION NO: N/A
11. TITLE (Include Security Classification) Storage capacity of the linear associator: Beginnings of a theory of computational memory	
12 PERSONAL AUTHOR(S) Mumme, Dean C.	
13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM 86Sept15 TO 91Sept14
14 DATE OF REPORT (Year, Month, Day) 1988 April 27	15. PAGE COUNT 123
16 SUPPLEMENTARY NOTATION	
17 COSATI CODES FIELD / GROUP / SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) connectionism, simulation, learning, memory, neural models

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

See reverse side

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b TELEPHONE (Include Area Code) (202) 696-4302 22c OFFICE SYMBOL N00014

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Storage capacity of the linear associator: Beginnings of a theory of computational memory

Dean C. Mumme
Learning Research and Development Center
University of Pittsburgh

This paper presents a characterization of a simple connectionist-system, the linear-associator, as both a memory and a classifier. Toward this end, a theory of memory based on information-theory is devised. The principles of the information-theory of memory are then used in conjunction with the dynamics of the linear-associator to discern its storage capacity and classification capabilities as they scale with system size. To determine storage capacity, a set of M vector-pairs called "items" are stored in an associator with N connection-weights. The number of bits of information stored by the system is then determined to be about $(N/2)\log_2 M$. The maximum number of items storable is found to be half the number of weights so that the information capacity of the system is quantified to be $(N/2)\log_2 N$.

Classification capability is determined by allowing vectors not stored by the associator to appear at its input. Conditions necessary for the associator to make a correct response are derived from constraints of information-throughput of the associator, the amount of information that must be present in an input-vector and the number of vectors that can be classified by an associator of a given size with a given storage load.

Figures of merit are obtained that allow comparison of capabilities of general memory/classifier systems. For an associator with a simple non- linearity on its output, the merit figures are evaluated and shown to be suboptimal. Constant attention is devoted to relative parameter size required to obtain the derived performance characteristics. Large systems are shown to perform nearest the optimum performance limits and suggestions are made concerning system architecture needed for best results. Finally, avenues for extension of the theory to more general systems are indicated.[1]

---

[1]This research was sponsored by the Army Research Institute, under Contract No. MDA903-86-C-0149 and Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research under Contract Nos. N-0014-86-K-0107 and N-0014-86-K-0678. Work submitted as Ph.D. thesis to the University of Illinois.

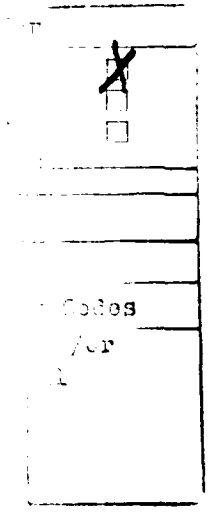
STORAGE CAPACITY OF THE LINEAR ASSOCIATOR: BEGINNINGS OF A THEORY OF COMPUTATIONAL MEMORY

Dean C. Mumme, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1988
Walter Schneider, Advisor

This thesis presents a characterization of a simple connectionist-system, the linear-associator, as both a memory and a classifier. Toward this end, a theory of memory based on information-theory is devised. The principles of the information-theory of memory are then used in conjunction with the dynamics of the linear-associator to discern its storage capacity and classification capabilities as they scale with system size. To determine storage capacity, a set of M vector-pairs called "items" are stored in an associator with N connection-weights. The number of bits of information stored by the system is then determined to be about $(N/2)\log_2 M$. The maximum number of items storable is found to be half the number of weights so that the information capacity of the system is quantified to be $(N/2)\log_2 N$.

Classification capability is determined by allowing vectors not stored by the associator to appear at its input. Conditions necessary for the associator to make a correct response are derived from constraints of information theory and the geometry of the space of input-vectors. Results include derivation of the information-throughput of the associator, the amount of information that must be present in an input-vector and the number of vectors that can be classified by an associator of a given size with a given storage load.

Figures of merit are obtained that allow comparison of capabilities of general memory/classifier systems. For an associator with a simple non-linearity on its output, the merit figures are evaluated and shown to be suboptimal. Constant attention is devoted to relative parameter size required to obtain the derived performance characteristics. Large systems are shown to perform nearest the optimum performance limits and suggestions are made concerning system architecture needed for best results. Finally, avenues for extension of the theory to more general systems are indicated.



A-1

Acknowledgements

Special thanks to the people on my Ph.D. committee, Walter Schneider, research-advisor, Michael Falman, committee chairman, Robert Stepp, Gerald DeJong, Sylvian Ray. Their views during my preliminary examination provided direction to this research. Thanks also to Geoffrey Hinton, James McClelland, and Edward Posner who allowed me to present my work at their research seminars. Comments and observations made by these people clarified several relevant issues. In addition, I am indebted to Rich Golden for a deeper understanding of connectionist systems and to James Staniaw who provided a broader perspective for the results of this investigation.

Most everyone owes their success to secretaries and other unsung heros/heroines. Most notable were the efforts of Cathy Rupp and Nelle Payne at the University of Pittsburgh, Learning Research and Development Center, who pulled my coals out of the fire on several occasions. Additionally, my greatest regard and appreciation is due the clerical staff who worked for the University of Illinois Department of Computer Science during my stay on the Urbana-Champaign campus. Many Ph.D.'s including myself owe their success to the efforts of these highly professional, caring individuals.

This research was sponsored by grants for Walter Schneider Sr. Scientist, University of Pittsburgh, Learning Research and Development Center, with the Army Research Institute, under Contract No. MDA903-86-C-0149 and Personnel and with Training Research Programs, Psychological Sciences Division, Office of Naval Research under Contract Nos. N-00140-86-K-0107 and N-00014-86-K-0678.

Dean C. Mumme
Moscow, ID

Preface

The approach of Minsky and Papert in their book *Perceptrons* [35] provided the motivation for this research. Their analysis of the perceptron introduced useful mathematical tools for understanding performance-limitations of "neural-based" systems. In addition, it charted and quantified these limitations and identified important areas for future investigation. As a result, the book *Perceptrons* identified issues of learning and performance that have continued to be of concern to Connectionist researchers even now that the challenge for multi-level learning algorithms has to some extent, been answered. The author believes that the mathematical tools developed by Papert and Minsky will themselves be useful for better understanding of connectionist architectures. In the author's view, the only short-coming of the work done by Minsky and Papert (and perhaps Rosenblatt as well) was their perspective. They treated the perceptron from a "computer" point-of-view. It was expected, for example, to determine whether or not a "retinal object" was "connected" even when the off-on state of a single "pixel" could determine the correct answer.

Most certainly, natural perception-systems don't work in this fashion. Indeed, they must determine the connectivity of objects *despite* inconsistencies or noise in the input-stimuli. This eliminates the possibility of "computations" whose result is affected by a single stimulus element. The proper perspective for these systems in the author's view is a probabilistic one in which the system's proper response is characterizable in some way but is robust to uncertain, degraded, incomplete, and even inconsistent information. The classifier identified in this work typifies just such a system and the forgone analysis should exemplify the proper viewpoint and methods for future investigations of systems of this nature. In this light, this work will have been of merit if it has identified issues valuable to future efforts and provides methods for analysis of perceptual/cognitive systems.

Table of Contents

Chapter 1
Introduction

The systems under consideration are an outgrowth of work done on self-organizing automata and perceptrons [35, 38] and later work in parallel associative memories, e.g. [21, 40]. Minsky and Papert in [35] had carried out rather extensive mathematical analysis on perceptrons revealing inherent limitations in the classes of problems they could solve. These systems were "learning" automata expected to classify input "stimuli" based on their past experience on "training" inputs. Minsky and Papert showed that multiple-stages of perceptrons were required for many problems of interest yet no training algorithm guaranteed to converge to a solution was known at the time for multi-level systems. They concluded in their book that the systems held little promise and subsequent investigation of perceptrons evaporated.

Eventually however, with more powerful computers to carry out simulations, and the development of several multi-level learning algorithms [9, 22, 36, 40, ch. 5-8], descendant offshoots of the perceptron have regained interest. Currently a variety of these automata exist and are known by names such as "Neural-nets", "Parallel Distributed Processors" (PDP networks), "Associative Memories". They are collectively called "connectionist architectures" and have been studied as self-organizing memories of perception [28] content-addressable memories, heirarchical knowledge bases, and classification systems [5, 6] models of human "neural-computation" [6, 18] of human task performance and attentional learning [41, 44] speech performance and natural language understanding [13, 40, ch. 18, 42].

These and other efforts have led to guarded optimism for the future of connectionist architectures as knowledge engines or as models of human intelligence. Capabilities and limitations of both task learning and performance have been demonstrated.[1] However, though many mathematical investigations (e.g. Barto [9], Golden [15, 14], Grossberg [19, 18], Kohonen [28]), have been conducted, including information-capacity studies (see Abu-Mostafa [1, 2], Amit [3, 4], Keeler [27], Little, et. al. [32], McEliece, et. al. [34]), there is much room for development of analytical understanding of the capabilities of these systems.

---

[1] Good introductory articles to the subject include the books [21, 40]. For an introduction to the mathematics of "connectionist" or "neural-based" systems, see [7, 40, ch. 9].

Development of connectionist memory systems in several forms has changed the concept of memory from **storage memory** to what the author calls **computational memory**. Digital and other local memories are examples of storage memory and have been supplimented by the distributed overlayed memory systems. The latter have more complex characteristics. Interference between items stored result in the capability of these systems to implicitly represent the regularities relationships among the items. Subsequently, computation and storage in the system are no longer distinct processes but integral aspects of the same phenomenon. These systems are "information engines" or "computational memory" rather than "information receptacles".

A formulation is needed of memory as a general mode of storage and computation. An information-theoretic approach appears most natural and promises to identify the essential features of memory operation. The purpose of this thesis is threefold:

1. **Analytical Models:** A germinal characterization of memory theory will be presented. The capabilities and limitations of any memory should then be expressible in terms of information flow. Resultant information-theoretic relations will provide the desired means of analysis and a framework for understanding any particular memory system as a member of the general class of computational systems.

2. **Relavant Issues:** Theory in 1 is used to identify major issues to be addressed for the understanding of storage memory. These issues include identification of "memory tasks", amount of information provided by the memory for the task, amount of information required by the task for a given amount of storage, the maximum number of items storable in the system with respect to the specified task, definition of memory load, memory load v.s. performance, identification of particular tasks useful to computation.

3. **Evaluation of quantitative performance:** Performance of the associator with respect to issues identified in objective 2 is quantified utilizing the theory from objective 1. First, storage-capacity is evaluated so that the notion of "memory-load" can be developed. Classification capabilities are then evaluated as the memory-load is increased. Architectural considerations and hardware tradeoffs are addressed, as well as performance degradation due to the introduction of non-linearities at the system-output. Finally, figures of merit are used to compare system performance with the optimal.

It is intended that this work will provide the proper context and starting point for further investigation of memory as a computational structure.

1.1. "Neural-based" systems

Matrix models of parallel distributed memories were derived as a simplistic model of brain cell computation. In the model, the output of each cell is a real number, y representing the deviation of the cell's firing frequency from some reference frequency. As such, y can be negative as well as positive. The inputs $\{x_1, x_2, \ldots, x_n\}$ to the cell are similarly real valued and each input, x_i has an associated coupling strength w_i to the cell which determines the effectiveness of that input on the cell output. The

cell determines its output by taking the weighted average of the inputs.

$$y = \frac{1}{n}\sum_{i=1}^{n} w_i x_i$$

where $(w_1, w_2, \ldots, w_n)$ is called the cell's "weight-vector". The matrix memory is constructed from a collection of these cells, each sampling the same set of inputs. If n_I is the number of inputs to the memory and n_O is the number of cells in the memory, the vector $\mathbf{x} \equiv (x_1, x_2, \ldots, x_{n(I)})$ of inputs when presented to the input of the system produces an output vector, $\mathbf{y} \equiv (y_1, y_2, \ldots, y_{n(O)})$ given by the relation $\mathbf{y} = \frac{1}{n_I} W\mathbf{x}$ where W is the matrix of coupling weights w_{ji} connecting the ith input to the jth cell [21, 28]. We note that each "cell" or "unit" is merely taking the dot-product between the input-vector and the unit's weight-vector.

To store information in this system, two sets of vectors called the input prototypes $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M\}$ and the output prototypes $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M\}$ are used. For each input prototype $\mathbf{f}_m$, the weights of the system are adjusted so that the $\mathbf{g}_m$ vector results at the system output when $\mathbf{f}_m$ is presented at the input. The system is then said to associate $\mathbf{f}_m$ with $\mathbf{g}_m$. For each $m=1,2,\ldots,M$, the matrix that is used to associate $\mathbf{f}_m$ with $\mathbf{g}_m$ (called the m^{th} association) is the *outer-product* $\mathbf{g}_m\mathbf{f}_m^T$ [21, p. 18]. To store the M associations, these M matrices are added to obtain:

$$W = \sum_{m=1}^{M} \mathbf{g}_m \mathbf{f}_m^T \tag{1.1}$$

The information for each association is distributed over the whole of W and therefore is overlaid with the information for the other associations. The resulting interference between associations increases with M, and ultimately limits the number of associations storable in the system.

In the case that $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M$ are mutually orthogonal, no interference exists. When $\mathbf{f}_k$ is input to the system, we have[2]

$$W\mathbf{f}_m = \frac{1}{n_I}\sum_{m=1}^{M} \mathbf{g}_m \mathbf{f}_m^T \mathbf{f}_k$$

$$= \frac{1}{n_I}\mathbf{g}_k \mathbf{f}_k^T \mathbf{f}_k$$

[2]The symbol || here refers to the "length" of a vector given by the euclidean norm.

$$= \frac{1}{n_I}|\mathbf{f}_k|^2\mathbf{g}_k. \qquad k = 1,2,\ldots,M.$$

The matrix produces a multiple of $\mathbf{g}_k$ when $\mathbf{f}_k$ is present at the input. If the $\mathbf{f}_k$ are chosen so that $|\mathbf{f}_k|^2 = n_I$ then $\mathbf{g}_k$ is reproduced exactly [6, p. 804, 21, p. 18].

We will be concerned with the case that the input prototypes are not orthogonal. Noting that $\mathbf{f}_m^T\mathbf{f}_k$ is the dot-product $\mathbf{f}_k\cdot\mathbf{f}_m$ we can rewrite the product $W\mathbf{F}_k$ as

$$W\mathbf{f}_k = \sum_{m=1}^{M} (\mathbf{f}_k\cdot\mathbf{f}_m)\mathbf{g}_m$$

Now the dot-product between two vectors is a measure of how well they "match" (assuming all vectors have the same length). The product $W\mathbf{f}_k$ is therefore a linear combination of the output-prototypes with the coefficient of $\mathbf{g}_m$ being proportional to how well $\mathbf{f}_m$ matches $\mathbf{f}_k$, $m = 1, 2, \ldots, M$. Since the input-prototype that best matches $\mathbf{f}_k$ is the vector itself, it follows that the output-prototype that has the largest coefficient in the linear combination is the vector $\mathbf{g}_k$. In the chapters that follow, the prototypes will be chosen randomly in such a way that they will be very nearly orthogonal to each other. Therefore, the dot-products $\mathbf{f}_k\cdot\mathbf{f}_m$ will be small for $m = 1, 2, \ldots, M$, $m \neq k$. This means that as long as there are not too many prototypes stored in the system, $\mathbf{f}_k\cdot\mathbf{f}_m\mathbf{g}_k$ will be the dominant term in the

)utput prototypes. We conclude that the linear-associator can be seen as a
particular, it produces an output vector that is a best match to the prototype
st-matches $\mathbf{f}_k$ (from among all the input-prototypes) is present at the input.
tput vector will have contributions from other output prototypes and so is not
a strict sense. When a better best-match computation is needed, a device
is used.

1.2. Auto-association

The systems described above are called "hetero-associators" because the "input prototypes" are distinct from the "output prototypes". That is, $\mathbf{f}_m \neq \mathbf{g}_m$. In fact the dimensionality of the input prototypes may differ from the dimensionality of the output prototypes as seen above. An "auto-associator" is similar to the hetero-associator except that the input and output dimensionalities are the same as are the input and output prototypes. That is $\mathbf{f}_m \equiv \mathbf{g}_m$ $m = 1, 2, \ldots, M$. After the weights are adjusted for storage of the M associations, retrieval occurs when a "damaged" input is presented to the system. The "damage" is due to noise in the input signal or the fact that the input may be specified incompletely. The output that results is passed through a non-linearity [6, 40, p. 61-65, 324-325] to limit

the growth of the size of the vector components. The output will be a better rendition of the proper input prototype provided the matrix is not overloaded (i.e. provided M is not too large).

Since the output is an improved version of the input, the signal can be fed back to the input of the system to obtain further improvement. The process is repeated several times until the vector stabilizes. the result is generally a highly improved version of the initial input. The limitation keeps the output vector from growing without limit and tends to force it to stabilize at or very near the proper prototype [6, 24]. Variations of the auto-associator include the "Hopfield net" [23, 24, 25], the "Brain-State-in-a-Box" or "BSB" model [6, 14], and the "Boltzmann Machine" [22].

From the perspective of memory systems, the difference between hetero- associators and auto-associators is that for the latter, the input signal provides *direct* information about the output. In the hetero-associator, the input serves only as an "address" or "approximate address" from which the proper output is to be retrieved. The auto-associator's input is both an address and a partial specification of the proper output. In any event, the auto-associator produces an output that is the prototype that best-matches the input vector. The algorithm degrades as the system stores more prototypes but should be an improvement on the hetero-associator for the same storage load.

In the chapters to follow, we will often study the performance of a best-match algorithm that takes as its input a vector produced at the output of a linear-associator. The best-match algorithm considered in the analysis is arbitrary but could just as well be an auto-associator. The auto-associator's stored prototypes would be identical to the linear-associator's stored output-prototypes. The analysis will be concerned with the conditions under which the linear-associator (first-stage) can produce an output vector "recognizable" by the best-match process (second stage). The best-match algorithm will have "recognized" the output of the linear-associator if the algorithm produces the output-prototype of the linear-associator that corresponds to the input-prototype of the associator that is most similar to the associator's input vector (see figure 1-1). In this configuration, the combination of the linear-associator and the best-match algorithm form a classifier. The linear-associator "translates" the input vectors of a form similar to the input prototypes into a form similar to the output-prototypes. The best-match algorithm (possibly an auto-associator) then selects the output prototype that most corresponds to the input to the combined system. Each input prototype corresponds to a vector that the system is most likely to "see" at the input or that is most representative of a class/category of input that is important to the system. The corresponding output prototype constitutes the system response and is of a form corresponding with the system's internal representation of the category. The combined system produces a particular output prototype corresponding to the category to which the system input belongs. Our concern is with the performance of the linear-associator. We will identify the conditions under which it will produce an output vector of high enough "fidelity" that the combined system can categorize its input.

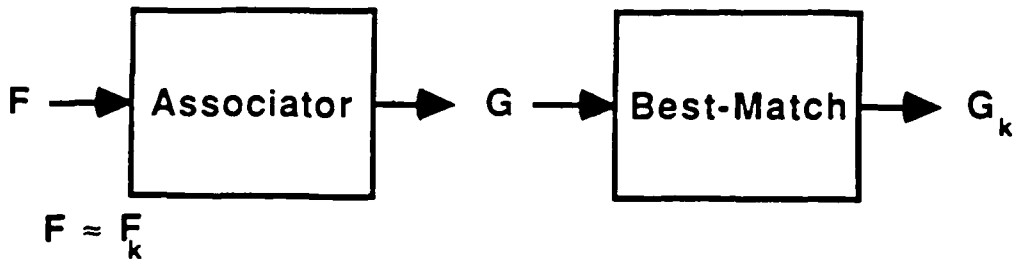


Figure 1-1: Linear-associator and Best-Match Classifier

Proper performance in this configuration is considered a minimal requirement on the linear-associator if it is to produce output "signals" useful to subsequent information-processing "stages".

1.3. Overview of Major Issues

1.3.1. Tasks of Computational Memory

The linear-associator is an example of "computational memory". As opposed to local memory which is merely an information storage device, computational memory is characterized as an input-output device that can respond to inputs that are not explicitly specified during storage. Similarly, the system can produce outputs not explicitly stored. The information stored in the memory is "overlaid" in the sense that all items (associations) stored share a common storage medium, resulting in between-item interaction of information. This interaction causes the output to be other than those explicitly trained to the memory. Instead the output is a function of how similar the input is to the trained inputs, and how similar the trained associations are to each other. This and the fact that the memory can respond to novel inputs results in a memory that is capable of various "memory tasks" during retrieval.

The most obvious (and mundane) of these is "item memory". For this task, the memory is treated just as a local-storage device by storing associations $(\mathbf{f}_m, \mathbf{g}_m)$, $m = 1,2,\ldots,M$ and subsequently using $\mathbf{f}_m$ as an "input address" to the memory which in turn returns information about $\mathbf{g}_m$ as "data". Another memory task is having the memory system distinguish which among the M output prototypes, is the one that matches the input prototype present at the input. Specifically, one first stores the

associations $(\mathbf{f}_m, \mathbf{g}_{\kappa(m)})$ where κ is a permutation of the M indices $1,2,\ldots,M$. One of the input prototypes, say $\mathbf{f}_k$ is then presented to the memory resulting in an output. This output is compared with all the output prototypes to identify one of the latter as a best match. The memory is successful at the task if $\mathbf{g}_{\kappa(m)}$ is the prototype chosen as the best match. This is called "channel-memory" since the memory acts analogously to a communication channel. Another term used is "permutation memory" indicating that the memory acts as a device that remembers which permutation κ of the output prototypes was associated to the input prototypes.

Though this task may seem artificial, its consideration serves two main purposes. First, proper performance of this task is a demonstration that the memory can distinguish the associations it has stored. If a system has stored too many associations, it may fail this task. If so, it is not providing enough information at the output to distinguish which prototype output was "intended" as the output of the memory. The stipulation that the memory succeed at this task is a minimal requirement called the "channel-criterion". The channel-criterion is used to derive upper bounds on the number of associations storable in the memory.

The second purpose for considering the matrix as a channel-memory is that we can then study the system performance with regard to the task of "input-classification". In particular, after the system has stored M association pairs $(\mathbf{f}_m, \mathbf{g}_m)$, non-prototype vectors are allowed at the memory input. Assuming that the input is most similar to the prototype $\mathbf{f}_k$, we will call the input vector $\mathbf{f}_k'$. To be successful classifying $\mathbf{f}_k'$, the matrix must generate an output that is most similar to $\mathbf{g}_k$. This is identical to the channel-memory task except that more freedom is allowed at the input. The classification task is important for understanding the system's ability to respond to a vector $\mathbf{f}_k'$ that is a partial or degraded (say, by noise) version of the "intended" input $\mathbf{f}_k$. The channel-criterion again provides a means of specifying limits on the number of associations storable in the memory for proper classification. In this case, a tradeoff is quantified between the number of associations permitted in the memory versus how "sloppy" $\mathbf{f}_k'$ can be as a rendition of $\mathbf{f}_k$. Consideration of the classification task allows one to identify the amount of information required by a linear-associator to classify an input-vector set of a given size into a given number of categories.

The classification task also brings up the issue of the reliability of the information at the output of the memory as a function of the reliability of the information presented to the memory input. This function depends on the number of associations stored in the memory. Storing more items taxes the memory capability and so requires that more reliable information be present at the input to maintain a given output reliability. An important issue is the determination of conditions necessary for the output information of the memory to be more reliable than the input information. Under such conditions, the memory could effectively supplment incomplete/degraded input information with its own stored

information to provide an output that is more complete/reliable. The memory task performed would be that of information "enhancement". An associator performing this task would be valuable as a "front end" to later stages of associator memories or processors that required "high-grade" information as input.

Even more intriguing is the possible use of this "enhancement memory" to iteratively improve the information it receives by passing the received information "through" the memory several times. Using two memory systems A and B, one stores associations $(\mathbf{f}_m, \mathbf{g}_m)$ in A and stores their *inverses* $(\mathbf{g}_m, \mathbf{f}_m)$ in B. One then sends an degraded copy $\mathbf{f}_k'$ of $\mathbf{f}_k$ to the input of memory A. The output of A is then input to B whose output is then *fed back* to the input of A. The process is then repeated. If both memories are "enhancement" devices, then the information that is passed back and forth between them should improve with each pass through the loop. Using the theory developed in this here, this possibility could be explored as a way to improve the performance of enhancement memories that have stored a given number of associations.

A final note concerning memory tasks is that they identify modes of "computation" that may serve as design tools for the architecture of connectionist "knowledge engines".

1.3.2. Characterization of Memory

Another important consideration is the definition of the "storage" of the memory. That is, defining the amount of information "contained" by the memory that is useful for retrieval. In particular, once M associations are stored, we consider the matrix $\mathbf{f}$ whose columns are the input prototype-vectors $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M$ and the matrix $\mathbf{g}$ whose columns are likewise the output-prototypes. For item memory discussed in the last section, the *storage* of the memory will be defined as the information that the matrix $\mathbf{f}$ provides about the matrix $\mathbf{g}$ via the memory. The question arises as to whether this is equal to the "item-information" which is simply the sum over $m = 1, 2, \ldots, M$ of the information that $\mathbf{f}_m$ provides about $\mathbf{g}_m$ via the memory. This work indicates an answer in the negative for linear-associative item-memory, under most conditions However, channel-memory does have this feature, again under most conditions. A memory having this feature will be called "item-accessible" meaning that essentially all the information that $\mathbf{f}$ provides about $\mathbf{g}$ via the memory can be retrieved "item-by-item". Like digital RAM memory (local storage), one can apply one input prototype at a time to the input of the memory and record the matrix output to retrieve all the information about $\mathbf{g}$. In fact, the information retrieved in this way is virtually non-redundant.

Characterization of memory as item-accessible allows upper bounds to be derived for the information retrievable from the application of a single input vector (called a single "access"). Since the system is symmetrically or uniformly defined over its input prototypes $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M$, the information

retrievable on applying any of these to the input is the same. From this it follows that the memory storage is just M times the amount of information retrievable from a single access just as is the case for local memory. The bounds that will be derived for the memory storage can thereby be mapped into bounds on the amount of information retrievable for a single memory access. Even for memory that is not item accessible however, the single-access bound will still hold. The difference is that the information retrieved by applying the M input vectors in sequence may "overlap" (redundancy) and as a result will not completely specify $\mathbf{g}$. We will characterize memory and address these issues after basic notions of information theory are introduced in the next chapter.

1.4. Methods and Focus of the Investigation

This investigation views the asymptotic performance of the linear-associator. That is, we examine the capabilities of the systems as they are allowed to get arbitrarily large. This will allow us to ascertain how well their performance scales with system size. Large systems benefit from the high diminsionality of their input/output signals and so perform better. Larger systems will therefore be most useful in memory/classification tasks and deserve the emphasis provided in this work.

The work is confined to finding upper bounds for system performance, though an effort is made to keep the bounds tight. Approximations are used extensively, but are accurate for the range of parameter-values considered. The approximations pertain particularly well to large-scale systems, with a correspondingly large number of associations stored. Pushing the lower limits of system size that the theory will accomodate, a system should have input/output dimensionalities of say 50 or 100 and at least 5000 weights. The number of associations should be at least 8 or 10 times the larger of the input/output dimensionalities, but generally no more than the number of weights in the system. More typically however, the input/output dimensionalities are taken to be at least several hundred each, and the number of items stored should be at least 25,000-50,000. The number of weights should generally be twice the number of stored associations or more.

In this work, an attempt has been made throughout to make explicit the range of applicability of the theory. The reader is advised to note parameter-value restrictions/assumptions made in what follows.

Chapter 2
Definitions, Identities and Notation

Before the presentation of memory theory, some preliminary material must be presented concerning the notation used and relationships that hold among information-theoretic quantities considered. More background concerning concepts of information theory can be found in texts [8, 12, 33].

2.1. General Relations of Information Theory

Unless otherwise stated, capitol letters always symbolize random variables whereas lowercase letters symbolize a specific value or random-variable outcome. Script-capitols represent sample-spaces. Within this convention, boldface *unsubscripted* letters represent matrices whereas boldface *subscripted* variables represent vectors. The letters $\mathbf{W}, \mathbf{F}, \mathbf{G}$ for instance, are random matrices; $\mathcal{W}, \mathcal{F}, \mathcal{G}$ are their respective sample-spaces; $\mathbf{w}, \mathbf{f}, \mathbf{g}$, represent respectively specific outcomes from each sample-space. Similarly $\mathbf{F}_m, \mathbf{G}_m$ are random vectors with respective outcomes $\mathbf{f}_m, \mathbf{g}_m$. The abbreviation "r.v." will be frequently used for "random variable" and the abbreviation "i.i.d." will be used for "independent, identically-distributed" when this condition applies to a random variable. The "equivalence sign", " $\equiv$ " will be used to denote "equality by definition" or the equivalence of two random variables. The random variables in this work are *discrete* with finite sample-spaces unless otherwise stated.

If $\mathcal{X}$ is the sample space for the r.v. X and for any $x \in \mathcal{X}$, $P(X = x)$ is the probability that $X = x$ then the **entropy** of X denoted $H(X)$ is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} P(X = x) \log_2 P(X = x)$$

If we define $p(x) \equiv P(X = x)$ then

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{2.1}$$

Heuristically, $H(X)$ is the average taken over all outcomes of X, of the minimum number of yes, no questions required to determine the outcome of X (see sections of [8, 12, 33] relevant to Huffman coding).

We call $H(X)$ the **uncertainty** of X, the **information content** of X or the **information represented** by X since it is the average amount of information required to determine X.

When considering two random variables X, Y the **conditional entropy** of X given Y is given by

$$H(X \mid Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(X = x, Y = y) \log_2 P(X = x \mid Y = y)$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the respective sample spaces of X and Y. This entropy can also be written

$$H(X \mid Y) = -\sum_{y \in \mathcal{Y}} H(X \mid Y = y) P(Y = y)$$

where $H(X \mid Y = y) = -\sum_{x \in \mathcal{X}} P(X = x, Y = y) \log_2 P(X = x \mid Y = y)$

The definition of entropy can be extended to n-tuples of r.v.'s $\mathbf{X}_n \equiv (X_1, X_2, \ldots, X_n)$. Examination of definition (2.1) reveals that $H(X)$ is not a function of the outcomes of X but of the probability function defined on those outcomes. In particular, X in equation (2.1) could be the vector-valued r.v. $\mathbf{X}_n$ or a matrix-valued r.v. $\mathbf{X}$. If the probability function P_n is defined over the sample space $\mathcal{X}_n$ of $\mathbf{X}_n$ then substitution of P_n for P in equation (2.1) gives

$$H(X_1, X_2, \ldots, X_n) = -\sum_{\mathbf{x} \in \mathcal{X}_n} P_n(X_1, X_2, \ldots, X_n = \mathbf{x}) \log_2 P_n(X_1, X_2, \ldots, X_n = \mathbf{x})$$

Note that $\mathbf{x} \in \mathcal{X}_n$ implies that $\mathbf{x}$ is an n-dimensional vector whose i^{th} component is a possible outcome of X_i. If $Y_1, Y_2, \ldots, Y_m$ is an m-tuple of r.v.'s, then we can extend the definition of conditional entropy to include $H(X_1, X_2, \ldots, X_n \mid Y_1, Y_2, \ldots, Y_m)$ which is the entropy of $X_1, X_2, \ldots, X_n$ conditioned on $Y_1, Y_2, \ldots, Y_m$ (see [8, 12, 33]). The important relationships are

1. $$H(X_1, X_2, \ldots, X_{n-1}) \leq H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i) \tag{2.2}$$

   where equality holds between the first and middle terms if and only if there is a function f so that $X_n = f(X_1, X_2, \ldots, X_{n-1})$ with probability one. Equality holds between the second and third terms if and only if the X_i's are mutually independent.

2. $$H(X_1, X_2, \ldots, X_n \mid Y_1, Y_2, \ldots, Y_m) \leq H(X_1, X_2, \ldots, X_n) \mid Y_1, Y_2, \ldots, Y_{m-1}) \tag{2.3}$$

   with equality if and only if $X_1, X_2, \ldots, X_n$ are independent of Y_m whenever the outcomes of $Y_1, Y_2, \ldots, Y_{m-1}$ are known.

3. $H(X_1,X_2,\ldots,X_n \mid Y_1,Y_2,\ldots,Y_m) \geq 0$ (2.4)

with equality if and only if $X_1,X_2,\ldots,X_n$ are completely determined by of $Y_1,Y_2,\ldots,Y_m$, that is, for each $i = 1, 2, \ldots, n$ there is a function f_i such that $X_i = f_i(Y_1,Y_2,\ldots,Y_m)$ with probability one.

Relation (2.4) holds when $m = 0$, that is

$$H(X_1,X_2,\ldots,X_n) \geq 0 \tag{2.5}$$

Particular inequalities implied by these relations are of concern, such as

$$0 \leq H(X \mid Y) \leq H(X) \leq H(X,Y) \leq H(X) + H(Y) \tag{2.6}$$

Equality holds respectively in each of the above inequalities if and only if $X = f(Y)$ with probability one; X and Y are independent; $Y = f(X)$ with probability one; X and Y are independent. Finally since we are only considering only discrete r.v.'s, for any deterministic function $f(x)$ we have

$$H(f(X)) \leq H(X) \qquad H(f(X) \mid Y) \leq H(X \mid Y) \tag{2.7}$$

$$H(f(X) \mid X) = 0 \tag{2.8}$$

$$H(Y \mid f(X)) \geq H(Y \mid X) \tag{2.9}$$

As remarked earlier, the entropy functions are functions of probability functions defined over sample spaces. Therefore the relations above hold even if the r.v.'s that appear in the expressions are scalar, vector, or matrix valued.

The **average mutual information** (or briefly "mutual information") between X and Y denoted as $I(X;Y)$ can now be defined

$$I(X;Y) = H(X) - H(X \mid Y) \tag{2.10}$$

It can be shown [12] that $I(X;Y)$ is symmetric in its arguments so that $I(X;Y) = I(Y;X)$. From this we also have

$$I(X\,;\,Y) = H(Y) - H(Y\,|\,X)$$

Also by equation (2.6) we have

$$I(X\,;\,Y) \geq 0 \tag{2.11}$$

with $I(X\,;\,Y) = 0$ if and only if X and Y are independent.

Consider again to the yes/no-question heuristic for guessing the value of X. Knowledge of Y is the equivalent of being provided answers to some of the questions required to determine X. This subsequently reduces the number of questions needed. The reduction given is precisely the uncertainty of X before Y is known minus the uncertainty of X after Y is known (i.e. identity (2.10)). We call this the information Y provides about X. By symmetry, this is also the information X provides about Y. As indicated in the previous paragraph, r.v.'s X and Y provide no information about each other if and only if they are independent.

If f is a deterministic function defined on the sample space $\mathcal{X}$ of X then then $H(f(X)\,|\,X) = 0$ and so

$$I(X\,;\,f(X)) = H(f(X)) \tag{2.12}$$

That is, the information X provides about $f(X)$ is precisely the information represented by $f(X)$. For any other r.v., Y, we have that $H(Y|\,f(X)) \geq H(Y|\,f(X), X) = H(Y|\,X)$ which implies $I(Y;\,f(X)) = H(Y) - H(Y|\,f(X)) \leq H(Y) - H(Y|\,X)$ and we have

$$I(Y;\,f(X)) \leq I(Y;\,X) \tag{2.13}$$

The concept of mutual information can be extended in ways analogous to the extensions of entropy outlined above. Two extensions concern us. First, the information $I(X\,;\,Y,\,Z)$ that two r.v.s Y and Z jointly provide about the r.v. X is defined by considering the pair $(Y,\,Z)$ as a single r.v. replacing the Y term in equation (2.10)

$$I(X\,;\,Y,\,Z) = H(X) - H(X\,|\,Y,\,Z) \tag{2.14}$$

Second, the information $I(X\,;\,Y|\,Z)$ that Y provides about X when Z is known is derived from the equation for $I(X\,;\,Y)$ by conditioning the entropies in (2.10) on Z

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) \tag{2.15}$$

A useful relation between $I(X;Y,Z)$ and $I(X;Y|Z)$ is

$$I(X;Y,Z) = I(X;Y|Z) + I(X;Z) \tag{2.16}$$

This can readily be shown by substituting for each term above its definition as a function of entropy.

We also need a fact used later about joint dependence. If W is a function of two r.v.'s X and Y jointly it is possible that W is independent of each of X and Y singly. That is

$$I(W;X,Y) = H(W) \tag{2.17}$$

$$I(W;X) = 0 \qquad I(W;Y) = 0 \tag{2.18}$$

An example is where X and Y are independent-identically-distributed (i.i.d.) r.v's; each takes values ± 1 with probability 1/2 that either value occurs. If $W \equiv X \cdot Y$, no information is conveyed about the outcome of W given only the outcome of X or given only the outcome of Y.

2.2. Specific Notation and Relations Required

2.2.1. Notation for Sets and r.v. Distributions

The symbol, $\mathbf{R}$, will be used in reference to the real-numbers. When speaking of a sequence of N entities a_n, $n = 1, 2, \ldots, N$, we will sometimes use the notation $\{a_n\}_{n=1}^{N}$. For infinite sequences, we substitute "∞" for N. Now let $\{X_n\}_{n=1}^{\infty}$ be a sequence of i.i.d. **Bernoulli** r.v.'s [30, p. 161], taking values $a, b \in \mathbf{R}$ with probabilities p and $(1-p)$ respectively. If Y_n is the sum of the first n Bernoulli r.v.'s, then Y_n is a **binomial** r.v. [30, p. 163] and we say Y_n is "$Bin(a,b,p,n)$" or more concisely, we put $Y_n \sim Bin(a,b,p,n)$. If $a = 1, b = -1$, and $p = 1/2$, then we put $Y_n \sim Bin(\pm 1, 1/2, n)$. Notice that in this case, the variance of Y_n is n. For a normal r.v., X with mean μ and variance σ^2, we put $X \sim N(\mu, \sigma^2)$. A normal r.v. with zero-mean and unit-variance is called a **standard normal** r.v. and "Φ" denotes the **standard normal distribution** function. The mean of an arbitrary r.v. X is denoted by EX and the variance by $VAR\,X$. The term, **random**, is used to refer to selection of an outcome of a uniform r.v. over a particular sample-space. The term **reliably** refers to an outcome or class of outcomes that occur with probability near one or with probability *approaching* unity as some relevant parameter gets large.

Most of the random vectors we consider will consist of ± 1's for components. We will call such vectors **$\pm$1-vectors** or **bit-vectors** since the components are binary. The set of n-dimensional bit-vectors is sometimes denoted by $\{-1, 1\}^n$ and often referred to as a "space" even though the set is not a proper vector-space over the real or complex numbers. If $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ is a random vector whose components X_i, $i = 1, 2, \ldots, n$ are i.i.d. each taking only the values ± 1, then $\mathbf{X}$ is called a **Bernoulli** vector. For the case that each of the two values ± 1 is taken with probability $1/2$, the vector $\mathbf{X}$ is called a **balanced-Bernoulli** vector. Note that choosing an n-dimensional balanced-Bernoulli vector is the same as choosing a vector at random from the n-dimensional space of bit-vectors.

2.2.2. Notations for Prototype-Vectors and the Associator Matrix

The vectors $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M$ and the vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M$ will be considered as outcomes of random input-vectors $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_M$ and random output-vectors $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_M$ respectively. The $\mathbf{F}_m$'s will be called **input-prototypes** and the $\mathbf{G}_m$'s will be called **output-prototypes**. These vectors are assumed to be balanced-Bernoulli vectors with n_I as the dimensionality of the input-prototypes and n_O as the dimensionality of the output-prototypes. We also form the random matrix $\mathbf{F}$ whose columns are $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_M$ in index-order. Similarly, we form the matrix $\mathbf{G}$ from the output prototypes. The symbols $\mathbf{f}$ and $\mathbf{g}$ of course denote particular matrix-valued outcomes of $\mathbf{F}$ and $\mathbf{G}$ respectively. The storage equation (1.1) becomes

$$\mathbf{W} = \sum_{m=1}^{M} \mathbf{G}_m \mathbf{F}_m^T \tag{2.19}$$

in terms of the random prototype-vectors. This can be expressed more concisely in terms of the matrices $\mathbf{F}$ and $\mathbf{G}$:

$$\mathbf{W} = \mathbf{G}\mathbf{F}^T \tag{2.20}$$

For retrieval, we form the matrix $\mathbf{G}'$ whose columns $\mathbf{G}'_k$ are given by

$$\mathbf{G}'_k = \mathbf{W}\mathbf{F}_k = \sum_{m=1}^{M} (\mathbf{G}_m \mathbf{F}_m^T)\mathbf{F}_k = \sum_{m=1}^{M} (\mathbf{F}_m \cdot \mathbf{F}_k)\mathbf{G}_m \tag{2.21}$$

or, in terms of the matrices

$$\mathbf{G}' = \mathbf{W}\mathbf{F} = \mathbf{G}\mathbf{F}^T\mathbf{F} \tag{2.22}$$

Another form of storage is called **channel-memory** or **permutation memory**. In this case, the output prototypes are considered to be known the retrieval device (later called the *detector*) and therefore will be denoted as specific outcomes $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M$. The input-prototypes $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_M$ will still be considered as random vectors. In addition, we will have need for the r.v. K whose outcome κ is one of $M!$ permutations of the indeces $\{1, 2, \ldots, M\}$. That is, κ is a function that maps any $m \in \{1, 2, \ldots, M\}$ to a unique value $\kappa(m)$ from the same set. This permutation is to be applied to the columns $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M$ of the g-matrix to produce the matrix $\kappa(m)$ whose columns are $\mathbf{g}_{\kappa(1)}, \mathbf{g}_{\kappa(2)}, \ldots, \mathbf{g}_{\kappa(M)}$. When considering the outcome κ of K as undetermined, we denote by $K(m)$ the r.v. whose outcome is the value $\kappa(m)$. The random matrix that results when κ is applied to $\mathbf{g}$ is denoted by $\kappa(\mathbf{g})$. Under these conventions the storage equation for permutation storage is

$$\mathbf{W} = \sum_{m=1}^{M} \mathbf{g}_{K(m)} \mathbf{F}_m^{\mathrm{T}} \tag{2.23}$$

or more concisely

$$\mathbf{W} = K(\mathbf{g})\mathbf{F}^{\mathrm{T}} \tag{2.24}$$

one says that the permutation K is stored in the memory.

2.3. Probabilistic Analysis of Sums

2.3.1. Distribution of Sums

Using the rightmost sum in equation (2.21), we can write the expression for the j^{th} component G'_{kj} of the random vector $\mathbf{G}_k$

$$G'_{kj} = \sum_{m=1}^{M} (\mathbf{F}_m \cdot \mathbf{F}_k) G_{mj}$$

$$= (\mathbf{F}_k \cdot \mathbf{F}_k) G_{kj} + \sum_{m=1;\, \neq k}^{M} (\mathbf{F}_m \cdot \mathbf{F}_k) G_{mj}$$

$$= n_f G_{kj} + \sum_{m=1;\, \neq k}^{M} (\mathbf{F}_m \cdot \mathbf{F}_k) G_{mj} \tag{2.25}$$

To extend the definition, we will have need for calculating the mean, variance, and entropy of such a sum. For this it will be useful to understand the independence of the terms under the summation.

To start, if $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are n-dimensional balanced Bernoulli-vectors with respective components X_i, Y_i, Z_i, then the dot-products $\mathbf{X} \cdot \mathbf{Y}$ and $\mathbf{X} \cdot \mathbf{Z}$ are independent. This follows from the fact that the products $X_i \cdot Y_i$ and $X_i \cdot Z_i$ are independent of their respective factors. In fact, this implies that $\mathbf{X} \cdot \mathbf{Y}$ is independent of $\mathbf{X}$ when $\mathbf{Y}$ is not known and vice-versa. Since the input-prototypes are balanced Bernoulli vectors, the dot products $\mathbf{F}_{m'} \cdot \mathbf{F}_k$ and $\mathbf{F}_m \cdot \mathbf{F}_k$ are independent when $m' \neq m$. Also the components of $\mathbf{G}$ are independent so the terms $(\mathbf{F}_m \cdot \mathbf{F}_k)G_{mj}$ in (2.25) are mutually independent.

Because of this independence, the variance of the sum is the sum of the variances of the summed terms. Furthermore, if two r.v.'s are independent with zero mean, then the variance of the product is the product of the variances. For each component X_i of an n-dimensional balanced Bernoulli vector $\mathbf{X}$, the mean EX_i is zero and the variance is one. Therefore, if $\mathbf{Y}$ is an independent n-dimensional Bernoulli vector the variance $VAR(X_i \cdot Y_i)$ is just $(VAR\,X_i)(VAR\,Y_i) = 1$. From this we have the variance

$$VAR(\mathbf{X} \cdot \mathbf{Y}) = VAR\Big(\sum_{i=1}^{n} X_i \cdot Y_i\Big) = \sum_{i=1}^{n} VAR(X_i \cdot Y_i) = n$$

From this we see that $VAR(\mathbf{F}_m \cdot \mathbf{F}_k)$ is n_I when $m \neq k$. Since the mean of G_{mj} is zero and the variance is one, we also have that the variance of $(\mathbf{F}_m \cdot \mathbf{F}_k)G_{mj}$ is n_I. These terms in the sum of (2.25) are independent and there are $M-1$ of them so the variance of the sum is $(M-1)n_I$. Considering the mean and variance of the $n_I G_{kj}$ term as well, we find that the mean of G'_{kj} is zero and the variance is Mn_I. The distribution of the sum on the right-hand side of (2.25) is $Bin(\pm 1, 1/2, M \cdot n_I)$ which is roughly normal. Considering the term $n_I G_{kj}$ again, we see that it takes values $\pm n_I$ with equal probability. We conclude that G'_{kj} is bimodal, each mode having a roughly normal distribution. Since $M-1 \approx M$ for large values of M the variance of each mode is taken to be Mn_I. Methods such as this are used in the chapter on classification to determine the distribution of sums.

2.3.2. Binomial Entropy

Another consideration is the entropy $H(S_n)$ of a sum S_n of n balanced Bernoulli r.v.'s X_i, $i = 1, 2, \ldots, n$. In the appendix it is shown that

$$H(S_n) = (1/2)\log_2(\pi e n/2) \tag{2.26}$$

Briefly the result is obtained as follows. First define a **standard Bernoulli** r.v. to be a r.v. that takes the value one with probability 1/2 and the value zero with probability 1/2. The sum S_n' of n standard Bernoulli r.v.'s is binomially distributed and takes on values s_n' that are in one-to-one correspondence with the possible values s_n of the sum S_n. To see this, note that the number s_n' is the number of summands of S_n' whose value is one. When the number of 1-valued summands of S_n is s_n' there will be $n - s_n'$ minus-1-valued summands of S_n. The value of S_n will therefore be $s_n \equiv n - 2s_n'$. This can also be written $s_n' = (n - s_n)/2$ completing the correspondence.

Under the one-to-one correspondence, S_n, and S_n' have equivalent probability distributions and so have the same entropy. Since the probability distribution of S_n' is determined by the binomial coefficients, we find the entropy of S_n' to get the entropy of S_n. Note that S_n' is binomially distributed and so is approximately normal with variance $n/4$. One might expect that the entropy of S_n' is approximately the same as that of a normal r.v. with the same variance. Appendix A shows that this is in fact true. That is, the entropy of S_n' is roughly $(1/2)\log_2(\pi en/2)$ where the approximation approaches perfection as n gets large. This of course implies that the entropy of S_n is $(1/2)\log_2(\pi en/2)$.

It is useful to note that although S_n is roughly normal with variance n, it does not have the same entropy as a normal r.v. with the same variance. Such a normal r.v. would have entropy $(1/2)\log_2(2\pi en) = (1/2)\log_2(\pi en/2) + 1$ which is 1 bit larger than the actual entropy of S_n. This discrepancy is due to the fact that we can multiply a discrete r.v. by any factor thereby changing its variance without changing its entropy. There is no strict correspondence between the variance and the entropy for discrete r.v.'s.

2.4. Special Functions

An entropy function of particular interest is the **binary entropy** function $\mathcal{H}(p)$. Let X be a r.v. with two outcomes x_1 and x_2 and probability p that x_1 occurs and probability $1 - p$ that x_2 occurs. Then

$$\mathcal{H}(p) \equiv H(X) = -p\log_2 p - (1-p)\log_2(1-p), \qquad 0 \le p \le 1 \tag{2.27}$$

Here $\mathcal{H}(0)$ is taken to be $\lim_{p \to 0} \mathcal{H}(p) = 0$. The function is continuous over the interval $[0, 1]$ and differentiable on $(0, 1)$.[3] It is strictly increasing on $[0, 1/2]$ and strictly decreasing on $[1/2, 1]$. By taking

[3] For real numbers $a \le b$, the *open* interval (a, b) is the set of real numbers between a and b excluding the endpoints. The *closed* interval $[a, b]$ includes the endpoints.

the Taylor series expansion of $\mathcal{H}(x)$ about $x = 1/2$ and truncating one can get an approximation of $\mathcal{H}(x)$ for $x \approx 1/2$. We also approximate $\Phi(x)$ for x near 0 in the same manner. These approximations are:

$$\mathcal{H}(x) \approx 1 - (2\log_2 e)(x - 1/2)^2 \qquad |x - 1/2| \leq 0.38 \text{ implies error} \leq 10\% \tag{2.28}$$

$$1 - \mathcal{H}(x) = (2\log_2 e)(x - 1/2)^2 \qquad \text{same error as above} \tag{2.29}$$

$$\Phi(x) \approx \frac{1}{2} + \frac{x}{\sqrt{2\pi}} \qquad |x| < 1 \tag{2.30}$$

2.5. Measuring Similarity

Just as storage of information is attributed to a "memory device" retrieval of the information is attributed to a "detection device" or **detector**. Both the memory and detector are characterized as mathematical processes. A particular mathematical process for the detector is that of measuring similarity between two vectors as is the case when the detector is a best-match process. The information retrievable by the detector will depend upon the similarity measure employed. Therefore, the performance of a system must be defined with respect to a particular similarity measure. We will define a *first order similarity measure* by way of the **Hamming-distance** function.

Definition 1: Define $\{-1,1\}^n$ to be the set $\{\mathbf{x} \in \mathbf{R}^n \mid x_i \in \{-1,1\},\ i=1,2,\ldots,n\}$. The **Hamming-distance** between two vectors is the function $HD:\{-1,1\}^n \times \{-1,1\}^n \rightarrow \mathbf{R}$ given by $HD(\mathbf{x},\mathbf{y}) = \frac{1}{2}\sum_{i=1}^{n} |x_i - y_i|$.

The Hamming Distance is the number of components at which $\mathbf{x}$ and $\mathbf{y}$ disagree. Its negative is a *prototypical* similarity measure on $\{-1,1\}^n$ from which the componentwise similarity measure is defined.

Definition 2: Componentwise Similarity Measure: If V is an n-dimensional vector-space, then a (componentwise) **similarity measure** is a function $S: V \times V \rightarrow \mathbf{R}$ having the following properties:

1. **Symmetry:** For all $\mathbf{x},\mathbf{y} \in V$, we have $S(\mathbf{x},\mathbf{y}) = S(\mathbf{y},\mathbf{x})$.
2. **Reflexively-Maximized:** For $\mathbf{x}, \mathbf{y} \in \{\mathbf{x} \in V \mid |\mathbf{x}| = 1\}$, $S(\mathbf{x},\mathbf{y})$ is maximized by $\mathbf{x} = \mathbf{y}$.
3. **Hamming-Consistency:** For vectors $\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{z} \in \{-1,1\}^n$, the Hamming-distance inequality $-HD(\mathbf{x}, \mathbf{y}) \leq -HD(\mathbf{w}, \mathbf{z})$ implies $S(\mathbf{x}, \mathbf{y}) \leq S(\mathbf{w}, \mathbf{z})$.

4. **First-Order Invariant:** If κ is a permutation of the indices $1,2,\ldots,n$ and $\kappa(\mathbf{x})$ is the vector whose components are the components of $\mathbf{x}$ permuted by κ then $S(\mathbf{x},\mathbf{y}) = S(\kappa(\mathbf{x}),\kappa(\mathbf{y}))$.

Under this type of similarity, $\mathbf{x}$ and $\mathbf{y}$ are to said to be more similar than $\mathbf{w}$, $\mathbf{s}$ whenever $S(\mathbf{x},\mathbf{y}) < S(\mathbf{w},\mathbf{s})$. Condition 3 requires the similarity measure to be consistent with the negative Hamming-distance similarity, $-HD(\mathbf{x},\mathbf{y})$ on $\{-1,1\}^n$. We allow the word "minimized" to be replaced by "maximized" in 2 provided that the second inequality in 3 is reversed. This results in a function that is *minimal* for similar vectors. The negative of a similarity function is therefore also a similarity function.

Examples of first-order similarity measures include those based on *Minkowski Metrics*. That is, the form $S(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|^p$ or its negative can be used. An inner-product can also be used, e.g. the *dot-product*, $S(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} x_i y_i$.

The notion of similarity presented here is meant to be "distance-based". In a vector space, two vectors of the same length will become similar if their distance (as determined by the appropriate vector-norm) is decreased. For vectors of a fixed length, this amounts to decreasing the angle between the directions of the two vectors. This corresponds to minimizing their dot-product. Distance-based similarity measures, particularly the dot-product, are especially relevant to the study of the associator. The output of the associator is based upon the similarity of the input-vector to the associator's input-prototypees as determined by the dot-product (see equation (2.25)).

We do not discuss detection or best-match processes in this investigation, but point out that they play a role in the considerations made in the analysis. When discussing information that one vector provides about another, we have assumed the information is distance-information. This characterization of information is consistent with the dynamics of most "neural-networks". Each cell or unit computes its output as a function of the dot-product similarity of the input-vector and the unit's weight-vector. The "computation" done by an associator is therefore based on similarity/distance information.

A best-match process used for detection (second-stage, as shown in 1-1) can itself be an associator or rather, an auto-associator and so will base its output upon distance-information relating the (first-stage) associator's output to the output-prototypes of the combined classifier. When speaking in later chapters of the information that the first-stage provides at its output, we will assume the information is distance/similarity information so as to be consistent with the nature of the best-match process. We also mention that the performance of a best-match process as a classification device will depend upon the similarity measure it uses. When comparing vectors, such a measure must preserve all distance information for optimal performance. We've assumed that distance information between two vectors is

completely specified by componentwise-similarity. Under this assumption, the dot-product seems optimal, at least for bit-vectors. When bit-vectors are to be compared, there is a one-to-one correspondence between the dot-product and the Hamming-distance so that the dot-product preserves Hamming-distance information.

Chapter 3
Information Theory of Memory

3.1. Introduction: Access v.s. Aggregate Memory

In this chapter a general information-theoretic formulation of memory is presented. Storage is characterized as the generation of a memory r.v. called the "memory trace" from two random variables called the address and the datum. Even if the memory trace is a deterministic function of the address and the datum, the address and datum are r.v.'s, so the memory state they generate during storage can be viewed as a r.v. from the point-of-view of retrieval. Retrieval is then the process of recovering information about the stored datum from the retrieval-address in the presence of the of the memory-state. The signal configuration for both storage and retrieval are specified allowing subsequent derivation of information-theoretic relations/limitations. These limitations are strongly dependent upon the retrieval strategy which may not utilize all information available from the memory. Retrieval methods will be formulated and performance of the system will be evaluated with respect to a particular retrieval strategy.

3.2. Information-Theoretic Characterization of Memory

3.2.1. Access v.s. Aggregate Retrieval

In this section we characterize memory as a configuration of r.v.'s and subsequently define memory retrieval. We show how information is stored/retrieved as an aggregate and then how it can be stored/retrieved as a collection of seperate datum-elements. The first of these modes is referred to as **aggregate-memory** and the second is **access-memory**. When an aggregate memory can be partitioned into access memory, we say that it is **accessible** and the storage (retrieval) of a datum-element is called a **storage-access** (**retrieval-access**).

For accessible systems, an upper bound is found for the **aggregate-information** the memory can provide and this is then used to upper-bound the amount of information the memory can provide during a single access (called **access-information**).

More explicitly, we have for aggregate memory the random variables called the **storage-address** $\mathbf{A}$ and the **storage-datum** $\mathbf{D}$. These are used during storage to generate the random variable $\mathbf{T}$ called the **memory trace** or simply the **memory**. During retrieval, the **retrieval-address** $\mathbf{A}'$ is used in conjunction with the memory trace $\mathbf{T}$ to obtain the **retrieval-datum** $\mathbf{D}'$. As a rule, the address r.v.'s $\mathbf{A}$ and $\mathbf{A}'$ must share information. That is $I(\mathbf{A};\mathbf{A}') > 0$ and from this one expects that during retrieval the memory will provide $\mathbf{D}'$ such that $I(\mathbf{D};\mathbf{D}') > 0$. As a rule, the larger the mutual information between $\mathbf{A}$ and $\mathbf{A}'$ is, the larger the mutual information between $\mathbf{D}$ and $\mathbf{D}'$ should be. For given r.v.'s $\mathbf{A}$ and $\mathbf{A}'$, the memory is optimal if $I(\mathbf{D};\mathbf{D}') = H(\mathbf{D})$. That is, the mutual information that the retrieval datum provides about the storage datum is maximized so that the retrieval datum completely specifies the storage datum.

For an aggregate memory to be accessible, it must have an **address-partition**. That is, there must exist r.v.'s $\mathbf{A}_m, \mathbf{D}_m, \mathbf{A}'_m, \mathbf{D}'_m$, $m = 1, 2, \ldots, M$, that partition $\mathbf{A}, \mathbf{D}, \mathbf{A}', \mathbf{D}'$ respectively so that $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_M)$, $\mathbf{D} = (\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_M)$, and similarly for $\mathbf{A}', \mathbf{D}'$. The storage and the retrieval processes must have partitions consistent with the address-partition. In particular, the memory trace $\mathbf{T}$ must be determinable from memory traces $\mathbf{T}_m$, $m = 1, 2, \ldots, M$; each $\mathbf{T}_m$ is generated exclusively from $\mathbf{A}_m$, $\mathbf{D}_m$. Similarly, the retrieval process should be capable of generating $\mathbf{D}'_m$ from $\mathbf{T}$ and $\mathbf{A}'_m$ alone. Also we require $I(\mathbf{A}'_m;\mathbf{A}_m) > 0$ and expect that retrieval produces a retrieval datum $\mathbf{D}'$ such that $I(\mathbf{D}'_m;\mathbf{D}_m) > 0$. In many cases (though not necessarily), optimal memory retrieval is taken to be the case in which each of the retrieval data $\mathbf{D}'_m$ completely specify each of the storage data $\mathbf{D}_m$.

We will make these notions more precise in the next section.

3.2.2. Formal Definition of Memory

Storage will be viewed as the generation of a **memory trace** $\mathbf{T}$ as a function of the **storage address** $\mathbf{A}$ and the **storage datum** $\mathbf{D}$:

$$\mathbf{T} = \mathrm{t}(\mathbf{A}, \mathbf{D}) \tag{3.1}$$

Retrieval is the subsequent generation of the **retrieval datum** $\mathbf{D}'$ as a function of the **retrieval address** $\mathbf{A}'$ and the memory trace $\mathbf{T}$[4]

[4]The memory trace $\mathrm{t}(\cdot)$ and the retrieval $\mathrm{d}'(\cdot)$ functions treated as *deterministic* in this development, hence the use of lower case letters t, d'. A more general formulation would allow the use of stochastic functions. However the deterministic case is pertinent to our situation and we deal with it specifically for the sake of simplicity. Note that a deterministic function of random variables produces a random variable.

$$\mathbf{D}' = \mathbf{d}'(\mathbf{T}, \mathbf{A}') \tag{3.2}$$

The **memory** is defined to be the quintuple $(\mathbf{A}, \mathbf{D}, \mathbf{A}', \mathbf{t}, \mathbf{d}')$. Notice that the memory trace and retrieval data are r.v.'s since they are functions of r.v.'s. The retrieval address is typically identical to the storage address or is a "degraded" version of it. We will generally consider the storage and retrieval address to be identical. If $\mathbf{A}, \mathbf{D}, \mathbf{A}', \mathbf{D}'$ and $\mathbf{T}$ are matrices, this retrieval process is equivalent to presenting the entire retrieval-address matrix $\mathbf{A}'$ to the memory to obtain the retrieval-datum matrix $\mathbf{D}'$ which in turn provides information about the entire storage-datum matrix $\mathbf{D}$. The aggregate-retrievable information $I(\mathbf{D}'; \mathbf{D})$ will therefore characterize the information that the memory can provide. For a given storage function for constructing $\mathbf{T}$, it is desirable to choose a retrieval function determining $\mathbf{D}'$ that maximizes $I(\mathbf{D}'; \mathbf{D})$.

3.2.3. Partitioning Memory: Formal Definition of Access-Memory

For access storage and retrieval, one partitions the storage address $\mathbf{A}$ and datum $\mathbf{D}$ into M parts $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_M$ and $\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_M$ respectively. For our situation the $\mathbf{A}_m$'s will be mutually independent and identically distributed over a common sample space and similarly for the $\mathbf{D}_m$'s. The storage process is in turn divided into M parts given by the relation

$$\mathbf{T}_m = \mathbf{t}_A(\mathbf{A}_m, \mathbf{D}_m), \qquad m = 1, 2, \ldots, M \tag{3.3}$$

The **access-storage function** $\mathbf{t}_A$ must be chosen so that $\mathbf{T}$ specified in (3.1) is a symmetric function $\mathbf{T} \equiv \mathbf{t}_S(\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M)$ of the $\mathbf{T}_m$'s. In other words, permuting the arguments of $\mathbf{t}_S$ doesn't change the value of the function determining $\mathbf{T}$.

The retrieval process is similarly divided into M parts. The retrieval address $\mathbf{A}'$ is partitioned into parts $\mathbf{A}'_1, \mathbf{A}'_2, \ldots, \mathbf{A}'_M$ and the retrieval datum $\mathbf{D}'$ into parts

$$\mathbf{D}'_m = \mathbf{d}'_A(\mathbf{T}, \mathbf{A}'_m), \qquad m = 1, 2, \ldots, M \tag{3.4}$$

The **access-retrieval function** $\mathbf{d}'_A$ must be chosen so that $\mathbf{D}'$ specified by (3.2) is the M-tuple $\mathbf{D}' = (\mathbf{D}'_1, \mathbf{D}'_2, \ldots, \mathbf{D}'_M)$. We call the quintuple

$$(\{\mathbf{A}_m\}_{m=1}^{M}, \{\mathbf{D}_m\}_{m=1}^{M}, \{\mathbf{A}'_m\}_{m=1}^{M}, \mathbf{t}_A, \mathbf{d}'_A)$$

the **access-partition** of the memory. A memory that has an access partition is called **access-memory**.

Under the conditions stated above, the information $I(\mathbf{D'}_m ; \mathbf{D}_m)$ that the m^{th} retrieved datum provides about the m^{th} storage datum should be independent of m. This hasn't been proven here, but the condition holds for memory systems we are interested in. We therefore assume that $I(\mathbf{D'}_m ; \mathbf{D}_m)$, called the **access-retrievable** information, is independent of m. The access-memory is said to be **access-separable** or **separable** if the r.v.'s $\mathbf{D'}$ and $\mathbf{D}$ and their respective partitions satisfy

1. **Access-Inclusive:** $I(\mathbf{D'} ; \mathbf{D}_m) = I(\mathbf{D'}_m ; \mathbf{D}_m) \qquad m = 1,2, \ldots, M$ (3.5)

2. **Access-Exclusive:** $I(\mathbf{D} ; \mathbf{D'}_m) = I(\mathbf{D}_m ; \mathbf{D'}_m) \qquad m = 1,2, \ldots, M$ (3.6)

3. **Access-Summable:** $I(\mathbf{D'} ; \mathbf{D}) = \sum_{m=1}^{M} I(\mathbf{D'}_m ; \mathbf{D}_m)$ (3.7)

If additionally, the value of $I(\mathbf{D'}_m ; \mathbf{D}_m)$ is the same for all m, then the memory information is said to be **uniformly access-separable** or simply **uniformly-separable**. In this case, for fixed m

$$I(\mathbf{D'} ; \mathbf{D}) = M \cdot I(\mathbf{D'}_m ; \mathbf{D}_m) \tag{3.8}$$

The first of the three conditions above states that the information that the m^{th} retrieval datum $\mathbf{D'}_m$ provides as much information about the m^{th} stored datum $\mathbf{D}_m$ as does the entire retrieved tuplet $\mathbf{D'} \equiv (\mathbf{D'}_1, \mathbf{D'}_2, \ldots, \mathbf{D'}_M)$. The idea is that $\mathbf{D'}_m$ *includes* all the information available about $\mathbf{D}_m$ that is available from $\mathbf{D'}$. Likewise, the second condition states that the information that $\mathbf{D'}_m$ provides about $\mathbf{D}$ is no greater than the information that it provides about $\mathbf{D'}_m$. Again, the idea is that $\mathbf{D'}_m$ *excludes* information about $\mathbf{D}_k$, $k \neq m$. Heuristically, the first condition states that $\mathbf{D'}_m$ provides *all* the information obtainable about $\mathbf{D}_m$ and the second states that it provides *only* information about $\mathbf{D}_m$. These two conditions would seem to imply the third, but the author has no proof for this. The conjecture, which could be false, is left here as an open question.

3.3. Characterization of Storage Capacity

3.3.1. Bounds on Retrievable Information

We now show that when the retrieval-address $\mathbf{A'}$ provides no direct information about the stored datum $\mathbf{D}$, the information, $I(\mathbf{D'} ; \mathbf{D})$, that the retrieval-datum $\mathbf{D'}$ provides about the storage-datum $\mathbf{D}$ is bounded by the storage-matrix entropy. Explicitly, we show

Theorem 1: Let $(\mathbf{A}, \mathbf{D}, \mathbf{A}', \mathbf{t}, \mathbf{d}')$ be a memory with $\mathbf{A}'$ independent of $\mathbf{D}$. Then

$$I(\mathbf{D}' ; \mathbf{D}) \leq H(\mathbf{T}) \tag{3.9}$$

Proof: Since $\mathbf{D}'$ is a function of $\mathbf{A}'$ and $\mathbf{T}$, we have by (2.13) that $I(\mathbf{D}' ; \mathbf{D}) \leq I(\mathbf{A}', \mathbf{T} ; \mathbf{D})$. By (2.16) we have

$$I(\mathbf{A}', \mathbf{T} ; \mathbf{D}) = I(\mathbf{T} ; \mathbf{D} \mid \mathbf{A}') + I(\mathbf{D} ; \mathbf{A}')$$

$$= H(\mathbf{T} \mid \mathbf{A}') - H(\mathbf{T} \mid \mathbf{D}, \mathbf{A}') \leq H(\mathbf{T})$$

where $I(\mathbf{D} ; \mathbf{A}') = 0$ since $\mathbf{A}'$ is independent of $\mathbf{D}$. The theorem follows.

We see from the proof of the theorem that

$$I(\mathbf{D}' ; \mathbf{D}) \leq I(\mathbf{A}', \mathbf{T} ; \mathbf{D}) \leq H(\mathbf{T}) \tag{3.10}$$

If $\mathbf{A}$ is independent of $\mathbf{D}$ then this relation holds for the case that $\mathbf{A}' \equiv \mathbf{A}$. If additionally, $\mathbf{A}$ is independent of $\mathbf{T}$ then the condition $\mathbf{A}' \equiv \mathbf{A}$ is optimal in that the second inequality of (3.10) becomes an equality. Since this will hold for the memory systems we consider, the relation will be displayed for future reference:

Corollary: When the conditions of theorem 1 hold for $\mathbf{A}' \equiv \mathbf{A}$ and $\mathbf{A}$ is independent of $\mathbf{T}$ we have

$$I(\mathbf{D}' ; \mathbf{D}) \leq I(\mathbf{T}, \mathbf{A} ; \mathbf{D}) = H(\mathbf{T}) \tag{3.11}$$

We now have a bound for the aggregate-retrievable information. If the memory is uniformly separable, then we will have a bound on the information retrievable on each access.

3.3.2. Storage and Storage Capacity

To obtain a bound on the information retrievable on the m^{th} access, assume that the memory $(\mathbf{A}, \mathbf{D}, \mathbf{A}', \mathbf{t}, \mathbf{d}')$ is uniformly separable. We then have for any $m = 1, 2, \ldots, M$:

$$M \cdot I(\mathbf{D}'_m ; \mathbf{D}_m) = I(\mathbf{D}' ; \mathbf{D}) \leq H(\mathbf{T}) \tag{3.12}$$

so that

$$I(\mathbf{D'}_m : \mathbf{D}_m) \leq H(\mathbf{T})/M \tag{3.13}$$

We will call this the **uniform-access bound**.

The uniform-access bound motivates the definition of storage and storage capacity for uniformly separable memory. For the systems we will consider, $\mathbf{A'} \equiv \mathbf{A}$ is optimal in the sense mentioned in the previous section. We assume then that the retrieval address is identical to the storage address and suppose that $I(\mathbf{D'}_m : \mathbf{D}_m)$ is independent of index m but is a function $I(M)$ of the number M of items stored. From (3.12), $I(M)$ must satisfy

$$M \cdot I(M) \leq H(\mathbf{T}) \tag{3.14}$$

The product on the left is the information storage of the system. The **storage capacity** will be defined as

$$C_S \equiv \max_M \; M \cdot I(M) \tag{3.15}$$

There are two ways to obtain a maximum of the number M of storable items. The first assumes that the product $M \cdot I(M)$ increases to a maximum as M increases to a value, M^*, then decreases. In this case equation (3.15) implies

$$C_S = M^* \cdot I(M^*) \tag{3.16}$$

where the right-hand-side is bounded above by the entropy $H(\mathbf{T})$ evaluated at M^* which we denote $H(\mathbf{T}, M^*)$. If $I(M^*)$ can be determined, then by (3.15)

$$M^* \leq \max_M \; H(\mathbf{T}, M^*) \; / \; I(M^*) \tag{3.17}$$

Another bound for M utilizes a lower bound $L(M)$ for $I(\mathbf{D'}_m : \mathbf{D}_m)$ as a criterion for system performance. Specifically, we make the constraint that

$$L(M) \leq I(M) \tag{3.18}$$

as a requirement for minimal system performance. If $L(M)$ is smaller than $I(M)$ for small values of M but overtakes $I(M)$ as M grows, a bound for M can be obtained from the constraint.

For the case that the memory is not separable, it may still be uniform in the sense that $I(\mathbf{D'}_m : \mathbf{D}_m)$ is independent of $m \in \{1, 2, \ldots, M\}$. For the instances we consider, relations (3.12) and (3.13) still hold so the methods of bounding M explained above apply. These methods will be utilized in the next chapter.

3.4. Relation of Separability of Memory to Performance

3.4.1. Non-Separability of Distributed Memory

For associative item-memory, we make the indentification $\mathbf{A}, \mathbf{A'} \equiv \mathbf{F}$, $\mathbf{D} \equiv \mathbf{G}$, $\mathbf{T} \equiv \mathbf{W}$ and $\mathbf{D'} \equiv \mathbf{G'}$. Aggregate storage is then given by (2.20) and aggregate retrieval by (2.22). The access-partition of the address and datum is just the division of the matrices into columns corresponding to the prototype vectors. The input-prototypes partition the address $\mathbf{F}$, each acting as a separate "address word" and the output-prototypes partition the stored-datum $\mathbf{G}$, acting as individual "datum words". The datum $\mathbf{G}_m$ is said to be stored at "location" $\mathbf{F}_m$. Access-storage is specified by (2.19) and access-retrieval is given by (2.21).

From calculations done outside this investigation, the linear-associator as an item memory is conjectured not to be separable except in limited cases. A preliminary development by the author has determined that item memory *might be* access-inclusive when $M \leq n_I/5$. Further, it may actually be separable when $n_I/5 \geq M \geq exp_2(n_O)$. These are submitted as sufficient conditions for separability but may not be necessary. A memory with an input-dimensionality exceeding $2 \cdot M$ and an output-dimensionality a few times $\log_2 M$ might be separable. Such a configuration is consistent with those considered later in the chapter on classification. For classification, systems with input-dimensionality greatly exceeding the output-dimensionality are most efficiently suited to the task.

On the other hand, separable memory is identical in function to digital RAM or local memory. The fact that matrix-based memories distribute the information for each association over the entire matrix means that the information for each association is *overlaid* with that of the others. This feature is what allows the information for separate associations to interact. Regularities in the input-to-output mappings specified by many associations should be "amplified" whereas irregularities/inconsistencies would be attenuated in the memory's input-to-output map. This interaction is contrary to the notion of separability. In fact, non-separability is the very feature that constitutes the capacity of distributed memory for "pattern discovery" [6, 40, ch. 1] and other functions that make them of computational interest. The non-separability of these systems makes their storage capacity more difficult to ascertain. However, the property "super-summable" exists for these systems so that bounds on the per-item

retrieval-information can be found in terms of the entropy of the matrix.[5] This results in a bound on the number of items storable in the system with respect to a minimal performance criterion.

3.4.2. Super-Summability of Item Memory

Assuming that item-memory is not separable, it may not be summable. However, the independence of the entries G_{kj} of the $\mathbf{G}$ matrix insures that the memory is **super-summable**. That is

$$I(\mathbf{G}' ; \mathbf{G}) \geq \sum_{k=1}^{M} I(\mathbf{G}'_k ; \mathbf{G}_k) \tag{3.19}$$

As we will see, this relation is quite useful in subsequent chapters on storage and classification. For the sake of later analysis then, we will start by showing this inequality and a useful extension of it hold. To start, $H(\mathbf{G}) = \sum_{m=1}^{M} H(\mathbf{G}_k)$ since the $\mathbf{G}_k$'s are independent. Also since $\mathbf{G} \equiv (\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_M)$ and $\mathbf{G}' \equiv (\mathbf{G}'_1, \mathbf{G}'_2, \ldots, \mathbf{G}'_M)$ we have that

$$H(\mathbf{G} \mid \mathbf{G}') \leq \sum_{k=1}^{M} H(\mathbf{G}_k \mid \mathbf{G}') \leq \sum_{m=1}^{M} H(\mathbf{G}_k \mid \mathbf{G}'_k)$$

always holds. Combining these, we get

$$I(\mathbf{G}' ; \mathbf{G}) = H(\mathbf{G}) - H(\mathbf{G} \mid \mathbf{G}')$$

$$= \sum_{k=1}^{M} H(\mathbf{G}_k) - H(\mathbf{G} \mid \mathbf{G}') \geq \sum_{k=1}^{M} \left(H(\mathbf{G}_k) - H(\mathbf{G}_k \mid \mathbf{G}') \right)$$

$$\geq \sum_{k=1}^{M} \left(H(\mathbf{G}_k) - H(\mathbf{G}_k \mid \mathbf{G}'_k) \right) = \sum_{k=1}^{M} I(\mathbf{G}'_k ; \mathbf{G}_k)$$

so that (3.19) holds. The extension of this is

$$I(\mathbf{G}' ; \mathbf{G}) \geq \sum_{k=1}^{M} \sum_{j=1}^{n_O} I(G'_{kj} ; G_{kj}) \tag{3.20}$$

which is proven in a similar manner by showing

---

[5] The term, "super-summable", is coined in analogy to the term "sub-sumable" used by mathematicians to describe non-linear functions $p(x)$ that obey $p(x+y) \leq p(x) + p(y)$. For our purposes, a "super-summable" function would have the inequality reversed.

$$I(\mathbf{G'}_k ; \mathbf{G}_k) \geq \sum_{j=1}^{n_O} I(G'_{kj} ; G_{kj}) \tag{3.21}$$

which holds because the components of $\mathbf{G}_k$ are independent.

The relations (3.19) and (3.20) are useful because $I(\mathbf{G'} ; \mathbf{G})$ is bounded above by $H(\mathbf{W})$ and so we have both

$$\sum_{k=1}^{M} I(\mathbf{G'}_k ; \mathbf{G}_k) \leq H(\mathbf{W}) \tag{3.22}$$

and

$$\sum_{k=1}^{M} \sum_{j=1}^{n_O} I(G'_{kj} ; G_{kj}) \leq H(\mathbf{W}) \tag{3.23}$$

Additionally, if the memory is uniform so that $I(\mathbf{G'}_k ; \mathbf{G}_k)$ is the same for all k, and $I(G'_{kj} ; G_{kj})$ is the same for all k, j, then (3.22) and (3.23) become

$$I(\mathbf{G'}_k ; \mathbf{G}_k) \leq H(\mathbf{W})/M \qquad k = 1, 2, \ldots, M \tag{3.24}$$

$$I(G'_{kj} ; G_{kj}) \leq H(\mathbf{W})/Mn_O, \qquad k = 1, 2, \ldots, M, \quad j = 1, 2, \ldots, n_O \tag{3.25}$$

Thus we get a bound on the information provided by any access-retrieval-data, $\mathbf{G'}_k$ about the storage-data $\mathbf{G}_k$ and also a bound on the amount of information any of the access-retrieval components G'_{kj} provide about the storage components G_{kj}.

These arguments hold when $\mathbf{G'}$ is replaced by some componentwise function $\mathbf{G''} \equiv \mathbf{g''}(\mathbf{G'})$ or rather $G''_{kj} \equiv g''(G'_{kj})$, as the retrieval function. The inequalities will be shown here for future reference

$$I(\mathbf{G''}_k ; \mathbf{G}_k) \leq H(\mathbf{W})/M \tag{3.26}$$

$$I(G''_{kj} ; G_{kj}) \leq H(\mathbf{W})/Mn_O \tag{3.27}$$

These bounds will be useful in later chapters on storage and classification.

3.4.3. Separability of Permutation Memory

For permutation memory, the storage address is the matrix $\mathbf{F} \equiv (\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_M)$. The $\mathbf{g}$-matrix in this equation is known to the detector and so is shown as a constant rather than a r.v. matrix. The storage-datum, $\mathbf{D}$, is a permutation r.v. K whose outcome κ is one of the $M!$ permutations of the indices $\{1, 2, \ldots, M\}$. That is, κ is a function that matches a given value m in $\{1, 2, \ldots, M\}$ with a unique value $\kappa(m)$ from the same set. To store the datum K, one applies K to the columns $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M$ of the matrix $\mathbf{g}$ to get the matrix, $K(\mathbf{g})$, whose columns are $\mathbf{g}_{K(1)}, \mathbf{g}_{K(2)}, \ldots, \mathbf{g}_{K(M)}$. The storage r.v. matrix is then obtained from $\mathbf{F}$ and K as in equation (2.24). The retrieval address $\mathbf{F}'$ is a matrix r.v. with $I(\mathbf{F}'; \mathbf{F}) > 0$. Often, we will take $\mathbf{F}'$ to be $\mathbf{F}$. The retrieval-datum, K' is a r.v. whose outcome κ' is determined as follows:

1. For $m = 1, 2, \ldots, M$, compute the vector $\mathbf{G}'_m = \mathbf{W}\mathbf{F}'_m$ and select via a similarity measure the vector $\mathbf{g}_k$ from among the output-prototypes that is a best-match of $\mathbf{G}'_m$. (In the case there is more than one such best-match, select one of them at random.)

2. Set $\kappa'(m) = k$.

This process represents the aggregate-retrieval function $\mathbf{d}'$. The access partition is the quintuple $(\{\mathbf{F}_m\}_{m=1}^M, \{K(m)\}_{m=1}^M, \{\mathbf{F}'_m\}_{m=1}^M, \mathbf{t}_A, \mathbf{d}'_A)$ where $\mathbf{t}_A$ is given by $\mathbf{t}_A(\mathbf{F}_m, K(m)) \equiv \mathbf{g}_{K(m)}\mathbf{F}_m^T$ and the access-retrieval function $\mathbf{d}'_A$ is calculated as shown in the two steps above for only one value of m at a time.

For *storage* of a permutation κ chosen randomly, the values $\kappa(1), \kappa(2), \ldots, \kappa(M)$ are nearly independent for large M. The only restriction on the $\kappa(m)$'s is that $\kappa(m') \neq \kappa(m)$ when $m' \neq m$. For large M, this restriction introduces little dependence among the values of $\kappa(m)$ $m = 1, 2, \ldots, M$. Since these M values are nearly independent, their joint entropy is approximately the sum of their individual entropies. The individual entropy is $\log_2 M$ bits, so the joint entropy roughly is $M \cdot \log_2 M$ bits. More precisely, the joint entropy is $\log_2 M!$ bits since the values $\kappa(m)$ specify one of $M!$ permutations. But $\log_2 M!$ is roughly $M \cdot \log_2 M$ for large M (say for $M \geq 3000$). Taking the values $\kappa(m)$, $m = 1, 2, \ldots, M$ to be independent is therefore a good approximation.

In the same way, *retrieval* of $K'(m)$ always gives *some* information about $K'(l)$ for $l \neq m$. This is because if the memory is accurate, then $K'(m) = K(m)$ with probability near one. Therefore, since $K(l) \neq K(m)$, the value of $K'(l)$ is not equal to $K'(m)$ again with probability near one. In short, knowing the value of $K'(m)$ gives "cross-over" information about $K'(l)$, $l \neq m$. In particular, the value of $K'(l)$ will probably not be the one observed for $K'(m)$. For accurate memory, we can compute

this cross-over information:

$$I(K'(m) : K'(l)) = H(K'(l)) - H(K'(l) \mid K'(m))$$

$$\approx \log_2 M - \log_2 (M-1) \approx 1/M$$

This is negligible compared to the uncertainty of $K'(l)$ for large M.

Due to the symmetry of the memory, and retrieval functions (the $\mathbf{F}_m$'s are i.i.d.) the probability $P(K' m) \neq K(m))$ is independent of m. Letting P_e be this probability, we seek the information $I(K'(m) ; K(m))$. To do so, we note that a best-match process that produces $K'(m)$ as its output, acts probabilistically as an M-ary symmetric communications channel [12] with $K(m)$ as the item to be "transmitted" and $K'(m)$ as the item produced at the "receiving end". We also have P_e as the probability of error at the receiving end. From this it follows that the information that the output provides about the input is given by

$$I(K'(m) ; K(m)) = \log_2 M - P_e \log_2 (M-1) - \mathcal{H}(P_e)$$

$$\approx (1 - P_e)\log_2 M - \mathcal{H}(P_e) \tag{3.28}$$

which is the information that the received signal provides about a transmitted signal that was sent over the communication channel. For small P_e, $I(K'(m) ; K(m))$ is approximately $\log_2 M$. On the other hand

$$\log_2 M \leq I(K'(m) ; K(m)) \leq I(K' ; K(m)) \leq H(K(m)) = \log_2 M$$

so that $I(K' ; K(m)) \approx I(K'(m) ; K(m))$ so the memory is access-inclusive.

To show that the memory is access-exclusive, the arguement is similar. Assuming P_e is small, knowledge of either $K(m)$ or of K tells us with high probability, what $K'(m)$ will be (namely the same value as $K(m)$). We have

$$I(K'(m) ; K) \approx H(K(m)) \quad \text{and} \quad I(K'(m) ; K(m)) \approx H(K(m))$$

so $I(K'(m) ; K) \approx I(K'(m) ; K(m))$.

To show the memory is access-summable, we retain the assumption that P_e is small so that K and K' will be identical with near-unity probability. This gives the relation

$$I(K' ; K) \approx H(K) = \log_2 M! \approx M \log_2 M$$

As mentioned earlier $I(K'(m) ; K(m)) \approx \log_2 M$ so

$$I(K' ; K) \approx \sum_{m=1}^{M} I(K'(m) ; K(m))$$

We have shown that the memory is access-separable. Uniformity follows from the fact that $I(K'(m) ; K(m)) \approx \log_2 M$ for all $m = 1, 2, \ldots, M$. In the low-error case then, the memory is uniformly separable. The question regarding how separable the memory is for larger error is a subject open for further investigation. Since P_e is independent of M, uniformity should hold even in the case that P_e is large. The author's conjecture is that greater error will degrade separability gradually and perhaps negligibly provided that $(1 - P_e)\log_2 M \gg \mathcal{H}(P_e)$.

3.4.4. Relation of Performance, Item-Memory and Channel-Memory

The notion of permutation-memory is merely a formulation of the memory's ability to keep track of which input-prototype is mapped to which output-prototype. For fixed outcomes $\mathbf{f}_m$ and $\mathbf{g}_m$, $m = 1, 2, \ldots, M$ of the prototypes and two random permutations, K and K', a matrix storing the associations $(\mathbf{f}_m, \mathbf{g}_{K(m)})$ should be different from the matrix storing the associations $(\mathbf{f}_m, \mathbf{g}_{K'(m)})$. The difference should be reflected in the response of the two matrices to a given input. For associative memory, the input will be some prototype $\mathbf{f}_k$. For the associative-classifier, the input will be some bit-vector $\mathbf{f}_k'$ that is closer to $\mathbf{f}_k$ than it is to the other prototypes. For either case, the matrix-output, call it $\mathbf{g}'_k$, should reflect which output-prototype, $\mathbf{g}_{K(k)}$ or $\mathbf{g}_{K'(k)}$, was associated with $\mathbf{f}_k$. If $(\mathbf{f}_k, \mathbf{g}_{K(k)})$ is stored, then $\mathbf{g}'_k$ should be closer to $\mathbf{g}_{K(m)}$ than to the other output-prototypes. Likewise for the case that $(\mathbf{f}_k, \mathbf{g}_{K'(k)})$ is stored. In either case, the matrix-output should provide an outside observer (a detector/best-match-process that has access to the output-prototypes) enough information to decipher which output-prototype is matched-up with $\mathbf{f}_k$ within the associator. In effect, the matrix-output must provide enough information about the proper output-prototype (e.g. $\mathbf{g}_{K(k)}$ for the first matrix and $\mathbf{g}_{K'(k)}$ for the second) to distinguish it from among the M alternatives. Of course, the permutation used is imaginary in the sense that we can relabel the output-prototypes so that the matrix is seen to store the associations $(\mathbf{f}_m, \mathbf{g}_m)$. With this convention, the output $\mathbf{g}_k'$ should provide the detector with enough information, that is, $\log_2 M$ bits, to allow a detector to decide which output-prototype is "$\mathbf{g}_k$".

In terms of the random vectors, $\mathbf{G}'_k$ has a mean determined by $\mathbf{G}_k$ but is independent of the *individual* prototypes $\mathbf{G}_m$, $m \neq k$, and so $\mathbf{G}'_k$ provides no information about any *individual* $\mathbf{G}_m$.

The information that $\mathbf{G'}_k$ provides about the output-prototypes to discern $\mathbf{G}_k$ from among the M alternative prototypes, should be largely due to the information it shares with $\mathbf{G}_k$. This must be at least $\log_2 M$ bits so

$$I(\mathbf{G'}_k : \mathbf{G}_k) \geq \log_2 M \tag{3.29}$$

would seem to be the necessary constraint on item-memory.

The problem is that $\mathbf{G'}_k$ may not be independent of the set $\{\mathbf{G}_m \mid m = 1, 2, \ldots, M, m \neq k\}$ as a whole, especially when $\mathbf{G}_k$ is known. Therefore the information it provides about the "correct choice" among the prototypes may be dispersed among all prototypes. The author has no precise formulation for this problem other than the definition of access-separability mentioned earlier. With access-separable memory, the information that $\mathbf{G'}_k$ provides about the output-prototypes is exactly the information it provides about $\mathbf{G}_k$ so that (3.29) would be a natural consequence of the present discussion.

Although item-memory appears not to be separable, our dilemma is resolved by the following observations. First, since

$$I(\mathbf{G'}_k : \mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_M) \geq I(\mathbf{G'}_k : \mathbf{G}_k)$$

the constraint (3.29) will assure that the left-hand member of the above relation is at least $\log_2 M$. Another consideration is the detector itself. We assume that it merely compares $\mathbf{G'}_k$ with each of the prototypes individually, and then compares the M results. No calculation involving $\mathbf{G'}_k$ with more than one prototype at a time is allowed. A detector of this sort should only be sensitive to information $\mathbf{G'}_k$ provides about individual prototypes. This information is zero for all prototypes except $\mathbf{G}_k$. Condition (3.29) will therefore be *necessary* for the detector. Of course, a more sophisticated detector which may not require this condition for reliable performance, may perform better than indicated in the subsequent chapters.

Chapter 4
Evaluation of Information-Storage Capacity

The analysis to follow is concerned with the case that the number, M, of stored associations is larger than the input dimensionality, n_I, so that the input vectors are linearly dependent and interference effects must be accounted for. In this case the output vector is only an approximation of the proper prototype output. Our concern is the number M of associations that can be stored in a matrix of a given size before the output becomes unrecognizable.

4.1. Characterizing Storage Capacity

To estimate the storage capacity of the matrix, we examine a system that has stored M associations $(\mathbf{f}_m, \mathbf{g}_m)$, $m = 1,2,\ldots,M$ for some M. The input-prototype vectors are n_I-dimensional and the output-prototypes are n_O-dimensional. For simplicity of analysis the prototypes will be balanced Bernoulli-vectors (see chapter 2, p. 15). All input-prototypes will then have $|\mathbf{f}_m|^2 = n_I$ and all output-prototypes will have $|\mathbf{g}_m|^2 = n_O$. To motivate the method of storage measurement, we make an analogy with digital memory. The address to the digital memory can be viewed as an input vector and the retrieved data as the output vector. A particular address vector and the data vector stored at the address location can be regarded as a vector-association pair. The number of bits represented by the data vector is the information the system provides upon performing the input-to-output association. For digital memory, the number of bits represented is the same as the number of bit-locations in the data vector and so is identical with the dimensionality of the data vector. **Storage** is defined in this chapter as the amount of information per association multiplied by the number of associations stored in memory. **Storage capacity** is the maximum storage the system can provide. In this case, the storage capacity is limited by the number of storage locations of the memory. Though the dimensionality of both the input and output vectors is specified in advance, the data items are not. That is, the number of items that can be stored is not determined by what they are. In effect, being able to *retrieve* data from the memory has no meaning unless we are able to *store* an *arbitrary* data set at the outset (ROM is no exception, when we consider all memory configurations possible before burn-in). In essence, the question "What is the storage-capacity of the memory?" has no meaning when one is considering a specific device whose identity and

input-to-output mapping is already determined/unchangeable. A burned-in ROM for is no longer a storage device, merely a retrieval device.

For the matrix memory, the storage is likewise given by the information-per-association multiplied by the number of associations stored. The dimensionality of the input and output prototypes are specified in advance, but the prototypes themselves are not. That is, we cannot assume specific values for the prototypes in the analysis to determine the storage capability of the system. Since the prototypes to be stored determine the values of the weights of the memory-matrix, the matrix is itself unknown. For this reason, the storage of the memory is not defined for a particular matrix but rather for a *class* of matrices all of the same size.[6] The class of outer-product matrix-associators of a given size is the set of all matrices that can be generated from balanced-Bernoulli vectors via equation (1.1). The discussion above indicates that an association is not considered to be stored in a particular matrix of the class unless it is explicitly included in the sum, (1.1) that determines the matrix.

The information-per-association for matrix memory can be characterized in several ways, two of which are considered here. The first called **item-memory** chooses an arbitrary $k \in \{1,2,\ldots,M\}$ and presents the k^{th} input prototype to the system. The matrix-output is then regarded as a probabilistic rendition of the k^{th} output prototype. On the average (over all matrices of the class), given M, the matrix-output will provide a certain amount of information about the prototype output and this is taken as the information provided by the association.

The second method, **channel-memory** or **permutation-memory**, acts analogously to an information channel. The k^{th} input is presented to the system and an output is generated. The latter is compared with each prototype-output vector via a similarity measure and the best match from the prototypes is chosen. To perform correctly, the system is expected to produce the k^{th} output prototype as the best-match. If the l^{th} output prototype is chosen, an error is identified with $l \neq k$. The probability of error averaged over the matrix-class is taken as the error probability for the associator as an M-ary symmetric channel (see section 3.4.3). The average mutual information between the output and input is thus defined. This average is considered as the information per association. For channel memory, we

---

[6] In fact, Hinton (personal communication) observed that an n by n identity matrix seems to have an exponential amount of "storage" since 2^n vector-pairs seem to be "stored". That is, using n-dimensional vectors of ± 1's, one selects one from among the 2^n possible. This vector is placed at the input of the system to retrieve the same vector at the output. More generally however, this can be done with an *arbitrary* matrix. Simply select a vector (address) of ± 1's, present it at the input, "digitize" the output into ± 1's and say that the resulting vector (data) is the one "stored" at that address. This would give all matrices *exponential retrieval* but there is no *storage process* that allows one to specify which addresses are to be known by the matrix and what datum is stored at each address. This illustrates that storage and retrieval are not to be confused as being the same. On the other hand, they are not independent of each other either. Reliable retrieval of a stored association or "item" will require, for the associator at least, that less than an exponential number of items be specified *during* the storage process.

define for each pair of positive integers (N, M) the **matrix channel** *of size N on M associations* It consists of the *ensemble* of all possible matrices with $n_I n_O = N$ that can be constructed from a set of M balanced-Bernoulli-vector prototype-pairs $(\mathbf{f}_m, \mathbf{g}_m)$, $m = 1, 2, \ldots, M$. Mathematically the ensemble acts as a communication-channel of information theory. Once a *particular* set of associations is chosen for storage, a particular matrix is selected from the ensemble via equation (1.1). This matrix is deterministic and is not itself a communication channel and its storage is not defined.

For both item and channel memory, the storage is the product of M and the information I represented by a single association. Initially, the storage $M \cdot I$ of the matrix increases proportionally with M. However the error probability increases with M as well so that the information-per-association I gradually decreases. For some value M^* of M, the information per association begins to diminish more rapidly than M increases. At this point, storing more associations decreases the total information storage of the system. For $M = M^*$, the system has reached its storage capacity.

The fact that the total retrievable information decreases eventually as M gets large is not proven in this work. In fact, this may not be the case. On the other hand, the channel memory provides a minimal criterion for memory performance. To perform well as a channel, a system need only produce an output that is more similar to the appropriate output-prototype than to the others. In effect, this demands only that the system be able to tell the stored associations apart. This seems a natural minimal capability since item-memory by contrast demands that the matrix actually "reconstruct" the appropriate output prototype. A system that can do this even with low fidelity of reproduction, can still perform well as a channel. The channel memory defines a lower limit allowable for the fidelity. Since fidelity deteriorates as more items are stored, we obtain a maximum number of useful associations that can be stored by the system. Channel memory then is crucial in determining the "absolute maximum" number of associations to be stored in a system.

4.2. Bounds on Storage Capacity

4.2.1. Restrictions on Relative Magnitudes of Parameters

The analysis that follows assumes important restrictions on the magnitudes and relative sizes of the parameters. These restrictions are in addition to any others derived later in this chapter.

We begin with the requirement that the input prototypes and the output prototypes be distinct vectors. With this, the number M of prototype-pairs must satisfy $\lceil \log_2 M \rceil \leq n_I$ and $\lceil \log_2 M \rceil \leq n_O$. However if each of these relations is an equality, the prototypes are already determined. The only thing that can vary is which input prototype is paired to which output prototype.

There are $M!$ ways to form the prototype pairs and so $M!$ ways to form the matrix. Therefore the matrix entropy is $\log_2 M! \approx M\log_2 M$ bits which is somewhat less than we will find it to be when the prototypes are randomly selected. The "entropy-degradation" caused by a fixed prototype-set, would seriously limit the amount the amount of information the matrix can provide at its output.

In order to ensure that the matrix entropy is not compromised, we must be able to choose either the input prototypes or the output prototypes (or both) at random. If the randomly chosen input-prototypes are to be distinct with high probability, we must have $2\log_2 M < n_I$ and if the output-prototypes are to be randomly chosen, we need $2\log_2 M < n_O$. These requirements ensure that sampling without replacement is virtually identical to sampling with replacement so that no duplicate selections occur. If at least one of these two requirements is met, the matrix-entropy should not be degraded.

More stringent requirements are needed if the prototype vectors are to be dissimilar to each other. This requirement is necessary for the output prototypes if a best-match algorithm is to match the output of the linear-associator with the correct output-prototype. A few errors in the matrix output should not confuse the best-match process as they would if the prototypes are too similar. The requirement is also necessary for the input-prototypes when the linear-associator is doing classification (see next chapter) and the inputs to the matrix are expected to be similar but not identical to an input-prototype. To meet the requirement, the dimensionality of a vector space from which prototypes are to be chosen cannot be too small. If two balanced-Bernoulli vectors are chosen from an n-dimensional space then the number of components that are identical between the two has average $n/2$ and standard deviation of $\sqrt{n}/2$. Since agreement of exactly $n/2$ components corresponds to orthogonality and most vectors will fall within 2 or 3 standard deviations of the mean, the prototypes will be highly orthogonal if the mean is large compared to the standard deviation. For this, n should be at least 100 or so.

To ensure dissimilar vectors one must also consider the number of prototypes to be chosen. The minimal distance occurring between two balanced-Bernoulli vectors from among M vectors chosen from n-dimensional space is roughly $n/2 - \sqrt{2\ln M}\cdot\sqrt{n}/2$ (see appendix B). In order that the two most similar prototypes be dissimilar, we require that the above minimal distance be nearly $n/2$. This will occur when $\sqrt{2\ln M}\cdot\sqrt{n}/2$ is small in comparison. As we shall see, the number M of prototype-pairs to be stored in the matrix should not exceed the number of weights in the matrix. If the matrix is square, this means M will not exceed n^2 where n is both the input and output dimensionality. For this maximal value of M we need $\sqrt{2\ln M}\cdot\sqrt{n}/2$ to be several times smaller than $n/2$. This sets a minimal bound on n. If we require at least an eight-fold difference between $n/2$ and $\sqrt{2\ln M}\cdot\sqrt{n}/2$, then n must be just over 1900 or larger. A four-fold difference produces a lower bound just under 400. In any event, the prototype dimensionality, both input and output, should be several hundred if an associator is to discriminate well between a large number of stored prototypes.

4.2.2. Matrix Entropy

As shown in the previous chapter, the amount of information retrievable from the matrix $\mathbf{W}$ is bounded above by its entropy $H(\mathbf{W})$. In this section, the matrix-entropy is estimated and used to ascertain the storage capacity of the matrix.

Given the M input-output prototype-pairs $(\mathbf{f}_m, \mathbf{g}_m)$, the matrix defined by equation (1.1) is seen as the sum of M outer-product matrices. The m^{th} outer-product or **association-plane** or **plane**, is completely determined by the $n_I + n_O$ bits of $\mathbf{f}_m$ and $\mathbf{g}_m$. Its ji^{th} component c_{ji} is the product $f_{mi}g_{mj}$, which takes values in $\{-1,1\}$. The m^{th} association-plane is not changed if both $\mathbf{f}_m$ and $\mathbf{g}_m$ are multiplied by -1. This indicates that the m^{th} plane represents at most $n_I + n_O - 1$ bits of information. In fact, the entries of any given row and column are enough to determine every other entry in the plane. To illustrate, examine the k^{th} row and l^{th} column and the entry $c_{ji} = f_{mi}g_{mj}$. These three entries (bits) c_{ki}, c_{kl} and c_{jl} determine c_{ji} so that the parity of these four numbers is even. The $n_I + n_O - 1$ entries that make up a particular row and column, are easily seen to be independent, so that $n_I + n_O - 1$ is also the lower bound on the information in a plane. We conclude that each association-plane represents *exactly* $n_I + n_O - 1$ bits. We mention also that the entropy of the association plane is the same even when the output (input) prototypes are fixed outcomes leaving only the input (output) prototypes as balanced-Bernoulli vectors. From this we have that the matrix-sum $\mathbf{W}$ of the association planes has the same entropy from the point of view of an external process that has knowledge of either (but not both) the set of input-prototypes or the set of output-prototypes.

When the association-planes are summed, information is lost. To assess the matrix entropy, note that each of the entries W_{ji} of the matrix is the sum of M "bits" $f_{mi}g_{mj}$, $m = 1,2,\ldots,M$. Therefore $W_{ji} \sim Bin(\pm 1, M, 1/2)$. As shown in appendix A, the entropy of W_{ji} is

$$H(W_{ji}) \approx \frac{1}{2}\log_2 \frac{\pi e M}{2} \tag{4.1}$$

As mentioned in the previous chapter, the entropy of a set of random variables is bounded above by the sum of the individual entropies (see equation (2.2)). Since there are N weights, where $N = n_I n_O$, and since the weights have identical entropies, the upper bound of $H(\mathbf{W})$ is obtained by multiplying the common weight-entropy $(1/2)\log_2(\pi eM/2)$ by N. The entropy $H(\mathbf{W})$ will obtain this upper bound if and only if the weights are independent. The assumption that the weights are independent is false for individual association planes. However the planes are independent and the bit-patterns in one plane will not generally be present in the others. For the sum of M such planes where M is large, the weight-independence assumption should provide a close approximation the the true matrix entropy when M is much larger than both n_I and n_O. We conclude then that

$$H(\mathbf{W}) \approx \frac{1}{2} M \log_2 \frac{\pi e M}{2} \tag{4.2}$$

is a good approximation when $M \gg n_I$ and $M \gg n_O$.

4.2.3. Bound on the Number of Items Storable

Consider the situation in which the k^{th} input-prototype, $\mathbf{F}_k$ is present at the input of the linear-associator and some process provides information about the k^{th} output-prototype $\mathbf{G}_k$ on the basis of what it sees at the memory output. If the average information it provides about $\mathbf{G}_k$ is I bits then from relation (3.12) of the previous chapter, we must have

$$M \cdot I \leq H(\mathbf{W})$$

Replacing $H(\mathbf{W})$ with its upper bound

$$M \cdot I \leq \frac{1}{2} M \log_2 \frac{\pi e M}{2}$$

so that

$$\frac{M}{N} \leq \frac{\log_2 M + \log_2 (\pi e/2)}{2 \cdot I}$$

We make the approximation $\log_2 (\pi e/2) \approx 2$ to get

$$\frac{M}{N} \leq \frac{\log_2 M + 2}{2 \cdot I} \tag{4.3}$$

In the case that the process at the output of the matrix is a best-match algorithm, the matrix is acting as a channel. By equation (3.28), page 32, we have

$$I = \log_2 M - P_e \log_2 (M-1) - \mathcal{H}(P_e)$$

where P_e is the probability that the best-match process chooses a prototype other than $\mathbf{G}_k$ as the one most closely resembling the matrix-output vector. For our purposes, $M - 1 \approx M$ and so

$$I \approx (1 - P_e) \log_2 M - \mathcal{H}(P_e) \tag{4.4}$$

Equation (4.3) becomes

$$\frac{M}{N} \leq \frac{1}{2} \frac{\log_2 M - 2}{(1-P_e)\log_2 M - \mathcal{H}(P_e)} \tag{4.5}$$

Our criterion for minimal channel performance is that $P_e = 0$ in which case $I = \log_2 M$. This gives the upper bound on M/N

$$\frac{M}{N} \leq \frac{1}{2} + \frac{1}{\log_2 M} \tag{4.6}$$

for perfect channel performance. When M is large, say $\log_2 M \geq 16$, the upper bound for M/N is only negligibly larger than 1/2. Therefore we define the **storage load** or **load**, L, of the system to be the ratio $2M/N$. A load of 1 corresponds to storing half as many prototype-pairs in the memory as there are weights in the matrix. For large systems (50,000 weights or more), a load larger than one precludes operation of the memory as a perfect channel.

4.2.4. Trading Storage with Error

To understand how the load trades with error rate P_e, we rewrite equation (4.5) as the quotient

$$\frac{M}{N} \leq \frac{1}{2} \cdot \frac{\log_2 M + 2}{(1-P_e)\log_2 M} \Big/ \frac{1}{1 - \mathcal{H}(P_e)/[(1-P_e)\log_2 M]}$$

letting $x = \mathcal{H}(P_e)/[(1-P_e)\log_2 M]$ and assuming this fraction is less than 1/3, we use the approximation $1/(1-x) \approx 1 + x$ to get

$$\frac{M}{N} \leq \frac{1}{2} \cdot \frac{\log_2 M + 2}{(1-P_e)\log_2 M} \left[1 + \frac{\mathcal{H}(P_e)}{(1-P_e)\log_2 M}\right]$$

$$= \frac{1}{(1-P_e)} \frac{1}{2} + \frac{1}{\log_2 M} \left[1 + \frac{\mathcal{H}(P_e)}{(1-P_e)\log_2 M}\right]$$

If we assume that $P_e \leq 1/2$ and that $2/(\log_2 M)^2$ is less than say 1/16, then when we multiply out the right-hand-side, we can ignore the $\mathcal{H}(P_e)/[(1-P_e)(\log_2 M)^2]$ term to get

$$\frac{M}{N} \leq \frac{1}{(1-P_e)} \frac{1}{2} + \frac{1}{\log_2 M} + \frac{\mathcal{H}(P_e)}{2(1-P_e)\log_2 M} \tag{4.7}$$

This approximation is good for $M \geq 2^6$ when $P_e \leq 1\ 2$. These restrictions ensure that the "z" term defined above is less than 1 3 which in turn ensures that the term we ignored to get relation (4.7) is small. If we allow P_e to be as large as 3 4, then we obtain a minimum value, 2^{12}, of M required for the validity of (4.7).

A simpler bound for M/N is afforded for $M \geq 2^{16}$. In this case, if P_e is less than 1 2, the term $(1 - P_e)\log_2 M$ is much larger than $\mathcal{H}(P_e)$ so that the latter can be ignored in relation (4 5). The relation then becomes

$$\frac{M}{N} \leq \frac{1}{(1 - P_e)} \frac{1}{2} + \frac{1}{\log_2 M} \tag{4.8}$$

Notice that this is the bound in equation (4.6) multiplied by the inverse of the "success rate" $(1 - P_e)$. The approximation is valid for more modest values of M when P_e is smaller than 1/2. Summarizing the analysis for larger systems, the number N of weights needed to store M associations for fixed P_e is $O(M)$. Allowing the load factor $L \equiv 2M/N$ to be larger than 1, say $L = 1/(1 - r)$, $0 < r < 1$, implies the error rate P_e will be at least as large as r.

4.2.5. Storage Limits for Item Memory

Now we turn our attention to item-memory. We assume that when the k^{th} input prototype is presented to the matrix, the matrix output is used *exclusively* to produce a bit vector that is as accurate a rendition of the k^{th} output prototype as possible. It is assumed that no information other than that provided by the matrix-output is allowed for production of the bit-vector. To be consistent with the other sections of this thesis, we denote the systems "rendition" of $\mathbf{G}_k$ as $\mathbf{G}_k''$. The term, I, in equation (4.3) is now $I(\mathbf{G}_k'';\mathbf{G}_k)$. For the case that $P(G_{kj}'' = G_{kj}) \approx 1$, $j = 1, 2, \ldots, M$, we have that I must be n_O bits and so

$$\frac{M}{N} \leq \frac{\log_2 M + 2}{2n_O}$$

Substituting $n_I n_O$ for N and rearranging, a criterion for n_I is found

$$n_I \geq \frac{2M}{\log_2 M - 2} \tag{4.9}$$

For large M (say $M \geq 16$) we can ignore the 2 in the denominator to get

$$n_I \geq \frac{2M}{\log_2 M} \tag{4.10}$$

Since the bit-error rate is near zero, $\mathbf{G}_k''$ should be virtually identical to $\mathbf{G}_k$. If a best-match is used to select the output-prototype that is nearest to $\mathbf{G}_k''$, then $\mathbf{G}_k$ will be chosen with near certainty. In other words, if we define P_e as the probability that $\mathbf{G}_k$ is *not* chosen then P_e should be near zero.

For this condition to hold, the memory must provide enough information at its output to act as a channel with no errors. Therefore relation (4.6) must be satisfied. Using this together with (4.9) and the fact that $N = n_I n_O$ one gets a lower bound on n_O

$$n_O \geq \log_2 M + 2$$

which is a minimal requirement to be made considering the parameter constraints discussed earlier in the chapter.

For illustration, we design a matrix to store $M = 50{,}000$ pairs. With this large number, relation (4.6) implies that at N is at least 100,000. The minimal value for n_I becomes about 5700 and the minimum for n_O is about 18. With these values, the number of weights becomes 106,200. We will compare this with the matrix parameters derived in the next section in which the system is allowed to make errors.

4.2.6. Item-Memory with Errors

Now consider the case that the components of $\mathbf{G}_k''$ each agree with their counterparts in $\mathbf{G}_k$ with probability noticeably less than 1. Assume that the probability that a pair G_{kj}'' and G_{kj} agree is independent of $j = 1, 2, \ldots, n_O$ and call this probability p_G. The probability of disagreement between a pair of components is $1 - p_G$ which is non-zero and so $\mathbf{G}_k''$ will contain a substantial number of bits that are in error. In this case, a best-match algorithm that compares $\mathbf{G}_k''$ with the output-prototypes will have a probability $P_e > 0$ that the wrong match is made.

The information that $\mathbf{G}_k''$ provides about $\mathbf{G}_k$ is bounded above by the information $\mathbf{G}_k'$ provides about $\mathbf{G}_k$ and bounded below by the sum $\sum_{j=1}^{n(o)} I(G_{kj}''; G_{kj})$ of the information that $\mathbf{G}_k$ provides on a bit-by-bit basis. The argument that this is a lower bound is similar to the argument given in the previous chapter to substantiate relations (3.19) and (3.20). The information that G_{kj}'' provides about G_{kj} is given by $(1 - \mathcal{H}(p_G))$. Using the above lower bound for I, this implies that relation (4.3) holds with I

replaced by $n_O(1 - \mathcal{H}(p_G))$. Assume that $p_G \leq 0.88$ so we can approximate $1 - \mathcal{H}(p_G)$ by $(2\log_2 e)(p_G - 1/2)^2$ as per equation (2.29). Inequality (4.3) becomes

$$\frac{M}{N} \leq \frac{\log_2 M + 2}{2n_O(2\log_2 e)(p_G - 1/2)^2} \tag{4.11}$$

For $M \geq \exp_2(16)$, we can ignore the 2 in the numerator on the right to get

$$\frac{M}{N} \leq \frac{\ln M}{4n_O(p_G - 1/2)^2} \tag{4.12}$$

We can get a lower bound on n_I by replacing N in (4.11) by $n_I n_O$ and rearranging

$$n_I \geq \frac{4M(\log_2 e)(p_G - 1/2)^2}{\log_2 M + 2} \tag{4.13}$$

Again, assuming $M \geq 50{,}000$ we can use (4.12) to get

$$n_I \geq \frac{4M(p_G - 1/2)^2}{\ln M} \tag{4.14}$$

which holds for larger systems. We assume that $p_G > 1/2$ since $\mathbf{G}_k''$ is supposed to be a better-than-chance rendition of $\mathbf{G}_k$. With this assumption the above relation can be expressed as an upper bound on p_G achievable by a given n_I

$$p_G \leq \frac{1}{2}(1 + \sqrt{n_I \ln M/M}) \tag{4.15}$$

Since p_G is less than 1, there is a non-zero probability P_e that $\mathbf{G}_k''$ will be mistaken for some prototype other than $\mathbf{G}_k$. If we assume that a best-match among the output prototypes is sought using the vector $\mathbf{G}_k''$ then the information $I(\mathbf{G}_k''; \mathbf{G}_k)$ must exceed that required to operate the best-match process. The information required for a best-match process with error rate P_e is given by (3.28) of the previous chapter and we can assure that $I(\mathbf{G}_k''; \mathbf{G}_k)$ is larger than this by requiring

$$n_O(1 - \mathcal{H}(p_G)) \geq (1 - P_e)\log_2 M - \mathcal{H}(P_e)$$

Assuming that $P_e \leq 1/2$ so that $(1 - P_e)\log_2 M \geq (1/2)\log_2 M$, we take M to be larger than 50,000 as usual. This allows one to ignore the $\mathcal{H}(P_e)$ term so that we have

$$n_O(1 - \mathcal{H}(p_G)) \geq (1 - P_e)\log_2 M$$

With the assumption that $1/2 \leq p_G \leq 0.88$ we use the approximation (2.29) to get

$$2n_O(\log_2 e)(p_G - 1/2)^2 \geq (1 - P_e)\log_2 M$$

which yields the reciprocal relations between the error probabilities

$$P_e \geq 1 - \frac{2n_O}{\ln M(p_G - 1/2)^2} \tag{4.16}$$

$$p_G \geq 1/2 + \sqrt{(1 - P_e)\ln M/(2n_O)} \tag{4.17}$$

To obtain a bound on the matrix size, n_O can also be expressed in terms of the other parameters:

$$n_O \geq \frac{(1 - P_e)\ln M}{2(p_G - 1/2)^2} \tag{4.18}$$

Note that relation (4.18) must hold for p_G to satisfy both (4.17) and (4.15) simultaneously. From (4.18) and (4.14) we have $N \geq 2(1 - P_e)M$ which is the same bound as given in (4.8) for M large. While n_I and n_O depend on p_G, their dependence is reciprocal so the matrix-size needed to store M items is not affected by p_G given a fixed P_e.

We use these relations to design a matrix that can store $M = 50{,}000$ items with a channel error $P_e = 1/2$ and a output-bit error $p_G = 3/4$. From relation (4.18) we obtain $n_O = 44$. From (4.14) we also have $n_I \geq 1156$, so that $n_I n_O \approx 50{,}900$. Again the matrix is one which "fans-in" to produce a highly reliable output under a large storage load. Notice that in accordance with $(1 - P_e) = 1/2$, this system is roughly half the size of the one designed earlier for "perfect" item retrieval.

Under any of the above circumstances, the number of weights needed for storage is $O(M)$. Allowing $P_e \geq 0$ allows an advantage with M increasing roughly proportional to $1/(1 - P_e)$, $(P_e \leq 1/2)$. If a bit-error $p_G < 1$ is allowed, then P_e must be specified to determine n_I and n_O as a function of M. Notice that relations (4.13), (4.14) and (4.18) imply that n_I can be made smaller when p_G is near $1/2$, whereas n_O must be made larger to meet the same storage requirements since the number of weights must satisfy relations (4.11) and (4.5). Requiring that the bits of $\mathbf{G}_k''$ to be accurate forces either M or n_O to

be small. That is, either the matrix must store few vectors (small ratio M/N) or the size $N = n_I n_O$ of the matrix must be due largely to n_I. Heuristically, the matrix must be able to gather a large amount of information at the input compared with the amount it supplies at the output. One would suspect that the information supplied at the output is a function of the information available at the input. This observation, which will be shown to be true in the next chapter, will be instrumental in deriving results regarding classification.

4.3. Storage Efficiency

Storage efficiency of a matrix will be defined as the matrix-storage divided by the information required to represent a matrix associator on M associations. We know that the number of bits stored by the matrix is the matrix entropy $H(\mathbf{W})$. To get the number of bits required to store the matrix, we examine equation (1.1) to ascertain the range of values that the weights can assume. This equation reveals that each entry (weight) in an outer-product matrix is the sum of M bits. The range of values of each entry is the set of integers between $-M$ and M. The extremes are realized whenever the bits for that entry all agree in value. Further, the entry will be be even if and only if M is even. It follows that the number of values an entry can assume is $M+1$. This means that N weights will require $N\log_2(M+1) \approx N\log_2 M$ bits for storage. We define the efficiency η by the matrix-entropy divided by the number of bits needed to represent the matrix

$$\eta = \frac{H(\mathbf{W})}{N\log_2 M} = \frac{(1/2)N(\log_2 M + 2)}{N\log_2 M} \approx \frac{1}{2} + \frac{1}{\log_2 M} \tag{4.19}$$

which is the upper bound for the ratio of M to N. In this case, the efficiency is asymptotically 1/2.

This is not the best we can do however. From the proof of the "tails lemma" in appendix A, page 100, the entropy $H(W_{ji})$ of a weight of the $\mathbf{W}$-matrix can be approximated by considering only $2r_M + 1$ of the most central values that the weight can achieve where $r_M = \lfloor \sqrt{2M\log_2 M} \rfloor$. This means that only these values occur often enough to represent a significant amount of the information represented by the weight. So we can ignore the more extreme values the weight might take and thereby only need roughly $\log_2(2\sqrt{2M\log_2 M}) \approx (1/2)\log_2(2M\log_2 M) + 1$ bits to store each weight.

Let M_0 be a positive integer representing the maximum number of associations to be stored in the matrix. If we restrict the weights to range in value from $-\lfloor 2M_0\log_2 M_0 \rfloor$ to $\lfloor 2M_0\log_2 M_0 \rfloor$ then when the number M of associations stored is no greater than M_0, the tails lemma prescribes the maximum number of bits of information lost by making the range restriction. The maximum information lost is given by the upper bound for ϵ in the tails-lemma which is $2\log_2 e\,(eM_0)$ (see (A.42), condition 2 and

related footnote, page 100). Assuming that this is the amount of information that is lost for each weight, the total lost for the entire matrix is no more than $2N\log_2 e/(eM_0)$ bits. If the matrix is required to lose no than r bits of information due to the weight restriction, then set M_0 equal to N/r so that the maximum information loss is $2N\log_2 e/(eN/r) = 2r\log_2 e/e \approx r$ bits. For the case that the load L is expected to be less than 1 (that is we don't intend to overload the matrix), we can set M_0 to be $N/2$ and will lose no more than one bit for the whole matrix by restricting the weights to the prescribed range.

The efficiency of this new system is again the matrix-entropy divided by N times the logarithm of the number of values permitted for each weight

$$\eta = \frac{(1/2)N(\log_2 M + 2)}{N((1/2)\log_2 (2M) + (1/2)\log_2 (\log_2 M) + 1)}$$

$$\approx \frac{\log_2 M}{\log_2 M + \log_2 (\log_2 M)} \qquad \text{for large } M \tag{4.20}$$

which is asymptotically near 1. Therefore, by simply truncating the range of the weights, we can for a fully loaded matrix, achieve a storage efficiency near unity while losing an insignificant amount of information about the matrix.

Chapter 5
Classification

5.1. Introduction

Whereas the previous chapter considered the linear-associator as a memory, the present chapter will treat it as a classifier. The classifier is merely a generalization of the memory in which the input-vectors are no longer constrained to be input-prototypes. In this case, input-prototypes are each a representative or "prototype" of a distinct category of vectors in the input-space. An vector from the input-space belongs to a category if it is closer, under the Hamming-distance metric, to the prototype of that category than to other input-prototypes. The input-prototype and its category have a *corresponding* output-prototype that represents the category in the output vector-space and the associator has stored the correspondence between the input and output prototypes. In this characterization, classification is similar to channel-memory (see figure 5-1). The input-vector by virtue of its membership in a particular category, has a corresponding output-prototype which is the category's corresponding output-prototype. Proper classification consists of associating the input-vector to an output-vector that is closer to the input-vector's corresponding output-prototype than to the other output-prototypes.

The analysis begins with the characterization of the linear-associator as a classification device. A non-linearity is applied to the associator-output to facilitate the analysis. Minimal requirements necessary for proper performance of the classifier are explained and we describe the associator's information characteristics relating to achieving these requirements. Methods of generating input-vectors are formulated and are eventually shown to be equivalent from the point-of-view of the associator. The information flow from input to output, called the "throughput" of the associator, is then quantified and related to performance capability of the associator. We will then be in a position to determine the minimal size of sub-vectors within input-vectors that act as "cues" for the input-vector category. We will also quantify the percentage of the input-space that is classifiable by the system. We then "revisit" storage capacity and quantify its degradation due to the use of the non-linearity at the associator output. Near the end of the chapter, the theory is illustrated with a few classifier designs and a discussion of important aspects of their operation. Finally, we derive some merit parameters for judging storage classification performance of the associator as it compares with the best theoretically possible.

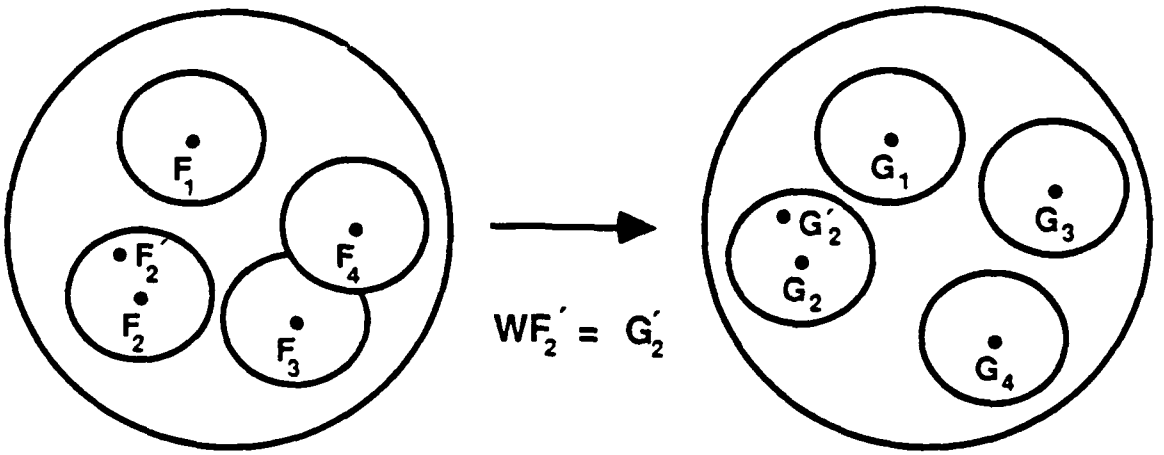


Figure 5-1: Classification by Prototype-Correspondence

5.2. The Associator as a Classifier

5.2.1. Characterization of Classification

Consider an arbitrary classification device as shown in figure 5-2. The device can receive any n_I-dimensional ± 1-vector as an input which will be referred to as the **input-vector**. The device has stored information about M vectors called input-prototypes. These prototypes are the n_I-dimensional balanced-Bernoulli vectors $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_M$. Each one is considered to be an **exemplar** of a distinct **category** of n_I-dimensional ± 1-vectors. An input-vector that is closest in Hamming-distance to the prototype $\mathbf{F}_k$ than to any of the other input-prototypes will be denoted by $\mathbf{F}_k'$ and is said to belong to the k^{th} category. Thus, there are M categories, each "centered" about its exemplar. After receiving the input $\mathbf{F}_k'$, the classifier is expected to emit the number k at its output to signal that the input belongs to category k. A **classification-error** (or briefly an "error") is said to have occurred when the response of the classifier is some number other than k. The probability of classification error is denoted P_e.

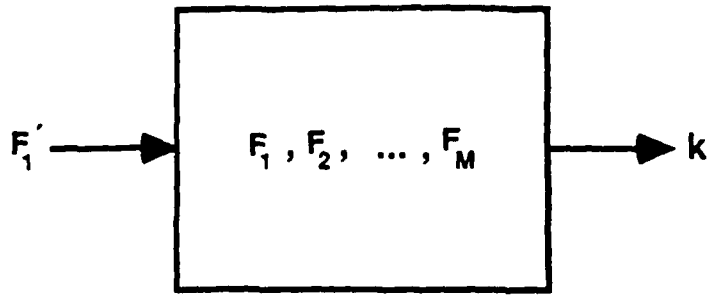


Figure 5-2: General Classifier for n_I-dimensional ± 1-vectors

If the classification device is to operate with negligibly small P_e, the input-vector, $\mathbf{F}_k'$ must provide at least $\log_2 M$ bits of information about its category-exemplar $\mathbf{F}_k$. This is due to the fact that $\mathbf{F}_k'$ must be distinguished as belonging to one of M categories and the only way the distinction can be made is to determine which of M exemplars is closest (see the chapter on the information-theory of memory). We therefore have the constraint

$$I(\mathbf{F}_k' : \mathbf{F}_k) \geq \log_2 M \tag{5.1}$$

Now consider the classification system of figure 5-3. In this case, the classifier is divided into two stages. The **first-stage** is a linear-associator whose output is fed to a **Hopfield-non-linearity** (defined later). This stage, called the **associator**, translates n_I-dimensional ± 1-vectors into n_O-dimensional ± 1-vectors where n_O is the dimensionality of the associator's output-prototypes $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_M$. The **second-stage** is a **best-match process** that compares the output of the first-stage with the output prototypes. In this case, the M category-exemplars for the classifier are the input-prototypes $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_M$. As is the case for the general classifier of figure 5-2, an input-vector that belongs to the k^{th} category will be denoted $\mathbf{F}_k'$. The resulting output of the linear-associator matrix will be called $\mathbf{G}_k'$ and the output of the Hopfield non-linearity is called $\mathbf{G}_k''$.

Upon receipt of $\mathbf{F}_k'$ at the input, the resulting vector, $\mathbf{G}_k''$, at the output is expected to be closer to $\mathbf{G}_k$ than to any other output-prototype. In this case, the best-match process of the second-stage process will respond with the number k at the output. We regard the best-match device as an error-free device. Errors will only occur if the first-stage produces a vector $\mathbf{G}_k''$ that is closer to some output-

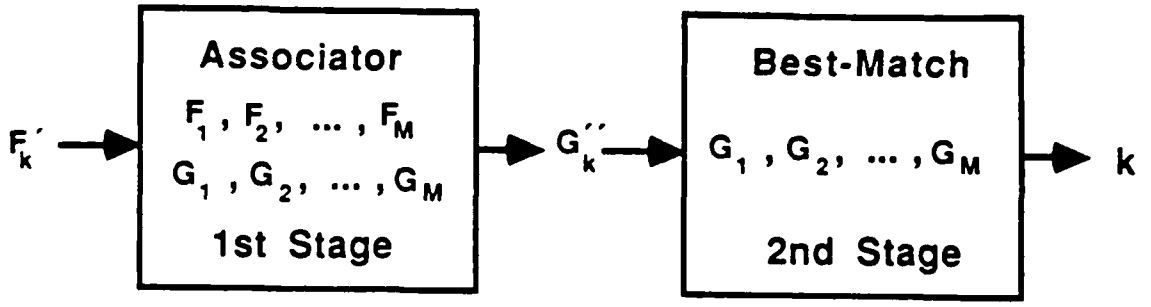


Figure 5-3: Associator Classifier for n_f-dimensional ±1-vectors

prototype other than $\mathbf{G}_k$. In other words, the analysis is concerned with the performance limitations of the first stage. The second-stage is merely an artifice for the sake of the characterization of the classification "task" of the linear-associator. In fact, the "classification" done by the associator is just its passing information to the output that enables one to determine which input-category is present at the matrix-input.

We observe that the second-stage of figure 5-3 is itself a classifier of an arbitrary sort. Its category exemplars are the vectors $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_M$ so its input $\mathbf{G}_k''$ must provide $\log_2 M$ bits of information about $\mathbf{G}_k$ if the second-stage is to classify reliably. The assumption that

$$I(\mathbf{G}_k'' ; \mathbf{G}_k) \geq \log_2 M \tag{5.2}$$

is thereby obtained as a constraint on the output $\mathbf{G}_k''$ of the first-stage.

In a later section it will be shown that the **output-information** $I(\mathbf{G}_k'' ; \mathbf{G}_k)$ of the first-stage can be regarded as a linear function of the **input-information** $I(\mathbf{F}_k' ; \mathbf{F}_k)$. The ratio $I(\mathbf{G}_k'' ; \mathbf{G}_k) / I(\mathbf{F}_k' ; \mathbf{F}_k)$ will be denoted by $T(\mathbf{W})$ and is called the **throughput** of the associator. Knowledge of the throughput will allow us to translate the constraint of (5.2) into a constraint on the input-vectors $\mathbf{F}_k'$. This in turn will reveal the fraction of the **input-space** $\mathcal{F}$ that can be classified. The general idea is to define the **input-redundancy** (or simply the **redundancy**) R of the input $\mathbf{F}_k'$ to

be the ratio

$$R = I(\mathbf{F}_k' \cdot \mathbf{F}_k), \log_2 M \tag{5.3}$$

The constraint (5.1) then stipulates that $R \geq 1$. The question is just how much redundancy must be present at the input to the associator to ensure reliable classification. The answer lies in the definition of throughput from which we have $I(\mathbf{G}_k'' : \mathbf{G}_k) = T(\mathbf{W})I(\mathbf{F}_k' : \mathbf{F}_k)$, and so relations (5.2) and (5.3) imply that the inequality $T(\mathbf{W})R\log_2 M \geq \log_2 M$ holds. That is

$$R \geq \frac{1}{T(\mathbf{W})} \tag{5.4}$$

In the case that the associator is not lightly loaded, $T(\mathbf{W})$ will be less than 1 so that by (5.3), the constraint (5.4) is more stringent than relation (5.1). Later it will be shown that at most M^{1-R} of the input-space $\mathcal{F}$ is classifiable. A heavily loaded associator will have a low throughput and so require a high redundancy. As a result, it can classify only a small portion of the input-space.

Since the classifier of figure 5-3 is merely an associator followed by a classifier, one may wonder why we should bother with the first-stage associator at all. One reason is that the associator translates input-vectors into output-vector "codes" that are more useful to subsequent processing stages. Another reason as we shall see, is the **data-compression** afforded by the associator. What data-compression is and its usefulness will be seen near the end of the chapter.

5.2.2. Generation of Input Vectors

An important aspect of associative memory is the ability to respond to input-patterns that deviate from the stored input-prototypes. In particular, suppose each input-prototype $\mathbf{F}_k$ is divided up into subvectors called **features** (see figure 5-4). That is, some subset of the n_I components of $\mathbf{F}_k$ represent a "field" in which a particular "piece" of information is coded. If $\mathbf{F}_k'$ has only this single piece of information in common with $\mathbf{F}_k$ and nothing (other than coincidental similarities) in common with the other input-prototypes, then we call $\mathbf{F}_k'$ a **single-feature vector**. It is desirable that an input-vector $\mathbf{F}_k'$ be classifiable even if it is a single-feature vector. Call the number of components of $\mathbf{F}_k$ that compose a particular feature the **feature-size**. We seek the minimal feature-size necessary for reliable classification of a single-feature vector.

Several methods of incorporating a feature of $\mathbf{F}_k$ in $\mathbf{F}_k'$ or inserting information about $\mathbf{F}_k$ into $\mathbf{F}_k'$ are considered here. The first is to copy r components of $\mathbf{F}_k$ into $\mathbf{F}_k'$ and set the rest of the components of $\mathbf{F}_k'$ to zero. This case can be reduced to analyzing the *storage* characteristics of an

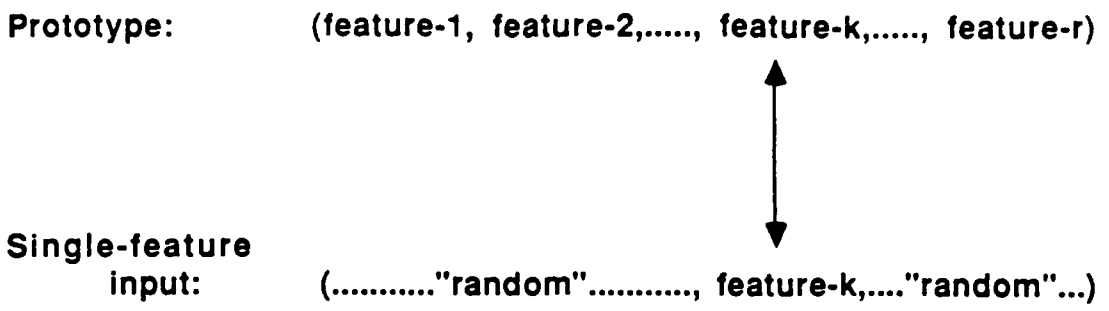


Figure 5-4: Features Within Vectors

associator with r-dimensional input. This method therefore is not as interesting as other methods which don't allow zeros as components of the input-vector. Zeroing the "unused" components however does have the advantage that no spurious information is incorporated into the input-vector. As far as the matrix is concerned, r bits of information are actually present at the input.

Another method is again to copy r of $\mathbf{F}_k$'s components to $\mathbf{F}_k'$ and choose the rest of $\mathbf{F}_k'$'s components as a random selection of ± 1's. This case is more interesting because it corresponds to $\mathbf{F}_k'$ containing information other than that of the r-dimensional feature of $\mathbf{F}_k$. This additional information however is not relevant to the prototypes of the associator. Rather, it is used by other associator-classifiers in a multi-classifier system (see figure 5-5). Each associator would sample the input-vector and only act on the features the input contains that are relevant to the prototypes of the associator. The input might represent the functional description of an object, each feature of the input-vector representing a different functional aspect of the object. Each associator would have information about a specific "feature-type" and associate features of this type to relevant "concepts" or "goals" of the system.

This method of generating the input-vectors actually incorporates r bits of information about $\mathbf{F}_k$ into $\mathbf{F}_k'$. However, the network is probably not capable of using all r bits of information. In the first place, the associator has no way of knowing which of the r of $\mathbf{F}_k'$ are the copies. What's more, it never varies the way in which it "weighs" a given component of $\mathbf{F}_k'$ when determining its output $\mathbf{G}_k'$. Whether or not it happens to weigh the r components of the feature heavier than the other components of the input, is a matter of "happenstance". Another related problem is that generating the input-vector with inconsistent information is not well-accounted for by information theory. An input-vector $\mathbf{F}_k'$

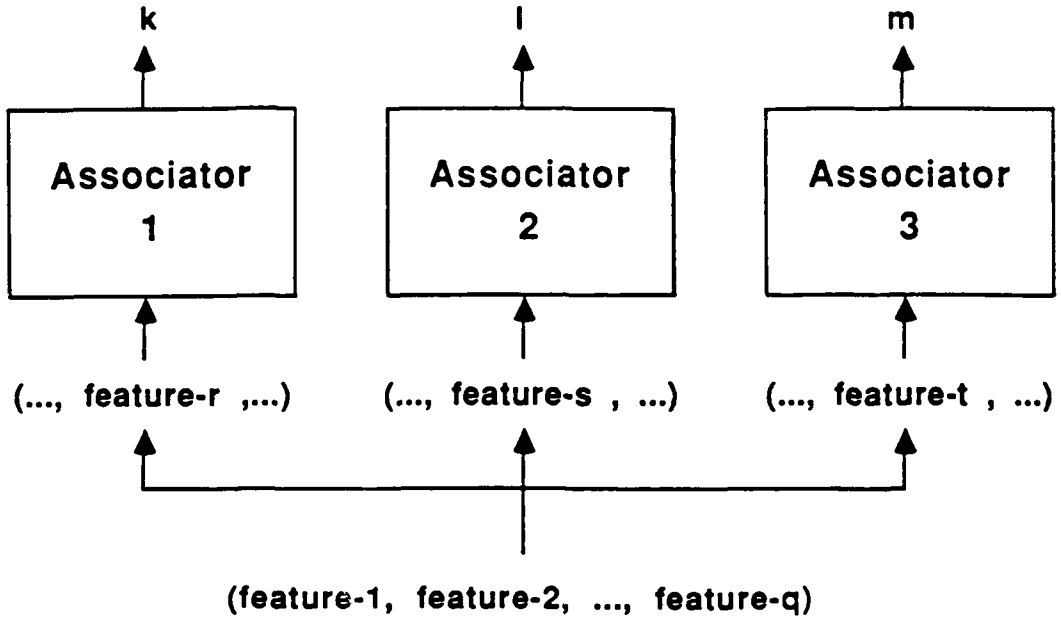


Figure 5-5: A Multi-Associator System

should be classified with the category-exemplar $\mathbf{F}_k$ even when it contains information in direct opposition to this choice of category. More precisely, copy r_1 components of $\mathbf{F}_k$ to $\mathbf{F}_k'$ and copy the negative of each of r_2 other components to $\mathbf{F}_k'$. Choose the remaining components of $\mathbf{F}_k'$ randomly. We assume $r_1 - r_2 > 0$ so that the *net* feature-size is $r \geq r_1 - r_2$. Again, if r is large enough, then the consistent information should "override" the inconsistent information so that $\mathbf{F}_k'$ is properly classified into the k^{th} category.

From an information-theory point-of-view however, the mutual information $I(\mathbf{F}_k'; \mathbf{F}_k)$ is no longer r bits but $r_1 + r_2$ bits. An observer of $\mathbf{F}_k'$, knowing which components were copied directly and which were negated could infer the $r_1 + r_2$ values of those components of $\mathbf{F}_k$. Of course, the associator treats all the components of the input-vector the same. If r is large, the dot-product $\mathbf{F}_k' \cdot \mathbf{F}_k$ of equation (5.12), page 57, will be large and $\mathbf{F}_k'$ will be correctly classified. From the point-of-view of the associator-matrix, the useful information is r bits not $r_1 + r_2$ bits. A more substantial argument for this

view will be given later. The arguement depends on the fact that the distribution of the matrix-output is a function of r only and does not otherwise depend on which of the above methods are used to generate the input-vector.

Another method of generating the input $\mathbf{F}_k'$ is to choose it within a region surrounding the prototype $\mathbf{F}_k$. We define the **ball** *of radius* ρ *about* $\mathbf{F}_k$ to be the set

$$\mathbf{B}_k(\rho) \;=\; \{\mathbf{x} \in \mathcal{F} \mid HD(F_k, \mathbf{x}) < \rho\} \tag{5.5}$$

where $HD(\mathbf{x}, \mathbf{y})$ is the Hamming-distance between the vectors $\mathbf{x}$ and $\mathbf{y}$. If $\rho > 0$ has a value such that $B_k(\rho) \approx 1/M$ then *conceivably*, each of the M balls $B_m(\rho)$ $m = 1, 2, \ldots, M$ could occupy its own region of the input-space $\mathcal{F}$ with little overlap. That is, most vectors of $\mathcal{F}$ would lie in exactly one ball. The likelihood of small overlap of all the balls is small but the important notion is that the largest portion of space each can occupy is $1/M$ without *unavoidable* overlap.

Now consider generating $\mathbf{F}_k'$ by choosing it at random from $B_k(\rho)$. We will call this method of input-generation the **neighborhood method**. An observer of $\mathbf{F}_k'$ knowing how it was generated, knows that the input-prototype $\mathbf{F}_k$ lies within ρ of $\mathbf{F}_k'$. Only $1/M$ of the input-space is this near $\mathbf{F}_k'$ so this knowledge constitutes an M-fold decrease in the number of possible values of $\mathbf{F}_k$. Therefore the vector $\mathbf{F}_k'$ chosen at random from $B_k(\rho)$ provides $\log_2 M$ bits of information about $\mathbf{F}_k$. Observe that if ρ were decreased so that $B_k(\rho)$ encompassed only M^{-R} of the space, where $R \geq 1$, then the input information $I(\mathbf{F}_k'; \mathbf{F}_k)$ would increase to $R\log_2 M$. This observation will be useful later when comparing the methods of generating the associator-input.

A final method of input-vector generation is that of flipping a biased coin to determine for each component (bit) of the input-vector $\mathbf{F}_k'$ whether it agrees with the corresponding component (bit) of $\mathbf{F}_k$. This will be referred to as the **coin** method. If the coin lands "heads", we copy a component of $\mathbf{F}_k$ to $\mathbf{F}_k'$. If it lands "tails", we copy its negative to $\mathbf{F}_k'$. Letting p_F be the probability of "heads", the probability that a component of $\mathbf{F}_k'$ agrees with its counterpart in $\mathbf{F}_k$ is p_F. In order that $\mathbf{F}_k'$ be a better-than-chance rendition of $\mathbf{F}_k$, we assume that $p_F > 1/2$. In this case, the information that $\mathbf{F}_k'$ provides about $\mathbf{F}_k$ is the sum over all n_I components of the information that each component of $\mathbf{F}_k'$ provides about its counterpart in $\mathbf{F}_k$. We can write

$$I(\mathbf{F}_k'; \mathbf{F}_k) \;=\; \sum_{i=1}^{n_I} I(F_{ki}'; F_{ki}) \tag{5.6}$$

The information $I(F_{ki}'; F_{ki})$ is the function $1 - \mathcal{H}(p_F)$ which is 1 bit minus the uncertainty $\mathcal{H}(p_F)$ that

F_{ki}' agrees with F_{ki}. When p_F is not too near 1. (say $p_F \leq 0.88$) we can approximate $1 - \mathcal{H}(p_F)$ by $2(\log_2 e)(p_F - 1/2)^2$ (see approximation (2.29) page 19). The result is

$$I(\mathbf{F}_k'; \mathbf{F}_k) = n_I(1 - \mathcal{H}(p_F))$$

$$\approx 2n_I(\log_2 e)(p_F - 1/2)^2 \qquad 1/2 < p_F < 0.88 \tag{5.7}$$

We can assess the similarity of the input-vector $\mathbf{F}_k'$ to the prototype $\mathbf{F}_k$ as measured by the dot-product. The average number of components of $\mathbf{F}_k'$ that agree with their counterparts in $\mathbf{F}_k$ is $n_I p_F$. The average number that disagree is $n_I(1 - p_F)$. The components that agree contribute a 1 to the value of the dot-product $\mathbf{F}_k \cdot \mathbf{F}_k'$ and the components that disagree add a -1. Therefore the mean of the similarity is

$$E(\mathbf{F}_k \cdot \mathbf{F}_k') = n_I p_F(1) + n_I(1 - p_F)(-1) = (2p_F - 1)n_I \tag{5.8}$$

For the method of copying r components to generate $\mathbf{F}_k'$, the mean similarity is r. We therefore set $r = (2p_F - 1)n_I$ to obtain the same mean similarity as for the coin method. This gives the reciprocal relations

$$r = (2p_F - 1)n_I \tag{5.9}$$

and

$$p_F = \frac{1}{2} + \frac{r}{2n_I} \tag{5.10}$$

It will be argued later that the various methods we described for generating the input-vector are equivalent, from the point-of-view of the associator, to the coin method with p_F given in (5.10).

5.2.3. Throughput of the Associator

To ascertain the throughput of the first stage of the classifier in figure 5-3 we must consider the probability distribution of the components of $\mathbf{G}_k'$. For $j = 1, 2, \ldots, n_O$, we show that the probability that $G_{kj}'' = G_{kj}$ is independent of j. Calling this probability p_G, it is shown to be a function of the probability p_F defined earlier. Consequently, the output information $I(\mathbf{G}_k''; \mathbf{G}_k)$, itself a function of p_G, is a function of the input-information $I(\mathbf{F}_k'; \mathbf{F}_k)$.

To assess p_G, note that $\mathbf{G}_k''$ is produced from $\mathbf{G}_k'$ via the "Hopfield" [24, 25] non-linearity

$$G_{kj}'' = \begin{cases} 1 & \text{if } G_{kj}' \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{5.11}$$

The probability that $G_{kj}'' = G_{kj}$ is the probability that $G_{kj}' \cdot G_{kj} \geq 0$ since the two relations are equivalent. As a result, we can compute p_G once the probability distribution of $G_{kj}' \cdot G_{kj}$ is known. Using the fact that $\mathbf{G}_k' = \mathbf{W}\mathbf{F}_k'$ where $\mathbf{W}$ is given by (2.19) we have

$$\begin{aligned} G_{kj}' \cdot G_{kj} &= \sum_{m=1}^{M} (\mathbf{F}_m \cdot \mathbf{F}_k') G_{mj} G_{kj} \\ &= (\mathbf{F}_k \cdot \mathbf{F}_k') G_{kj}^2 + \sum_{m=1, m \neq k}^{M} (\mathbf{F}_m \cdot \mathbf{F}_k') G_{mj} G_{kj} \end{aligned} \tag{5.12}$$

Using methods outlined in the chapter on notation, page 16, the probability function of the term $(\mathbf{F}_k \cdot \mathbf{F}_k') G_{kj}^2$ in (5.12), call it the "first term", can be determined. The same can be done for the summation (call it the "second term") in (5.12). Both the first term and the second term are sums of i.i.d. r.v.'s so that the central limit theorem implies the two are both normally distributed. The sum of two independent normal r.v's is normal so we conclude that $G_{kj}' \cdot G_{kj}$ is normal. The mean of $G_{kj}' \cdot G_{kj}$ is the sum of the means of the first and second terms of (5.12) and similarly for the variance. Recalling that $\mathbf{F}_k'$ is generated by the coin method with $p_F = 1/2 + r/(2n_I)$, the mean of the first term is $n_I(2p_F - 1)$ and the variance is $4p_F(1 - p_F)$. The mean of the second term is zero and the variance is $(M-1)n_I$. Therefore the mean of $G_{kj}' \cdot G_{kj}$ is $n_I(2p_F - 1) = r$ and the variance is $4n_I p_F(1 - p_F) + (M-1)n_I$. The latter is very nearly equal to Mn_I for any value of p_F provided $M \geq 10$.

Before calculating p_G in terms of p_F, we make some observations with regard to the effect of generating $\mathbf{F}_k'$ on the distribution of $G_{kj}' \cdot G_{kj}$. When $M \geq 10$, the variance of $G_{kj}' \cdot G_{kj}$ is determined entirely by the second term of equation (5.12). The balanced-Bernoulli vectors, $\mathbf{F}_m$, $m \neq k$, appearing in the second term are independent of $\mathbf{F}_k'$ regardless of how $\mathbf{F}_k'$ depends on $\mathbf{F}_k$ (see chapter 2, page 16, concerning dot-product independence). Thus the mean and variance of $\mathbf{F}_m \cdot \mathbf{F}_k'$ will not not be affected by any of the methods of generating a ± 1-vector $\mathbf{F}_k'$ from $\mathbf{F}_k$. From this we see that the variance of the second term will always be $(M-1)n_I$ irrespective of the method of generating $\mathbf{F}_k'$. Since $\mathbf{F}_k$ is a ± 1-vector, the variance of the first term of (5.12) can never exceed n_I. The first term will therefore not contribute substantially to the variance of $G_{kj}' \cdot G_{kj}$ under any method of input-generation. Also $G_{kj}' G_{kj}$ is normally distributed since the second term is a large sum of i.i.d.

r.v.'s. The nature of the first term is inconsequential due to its small variance. Further the mean of $G_{kj}'G_{kj}$ is r for any of the methods given for generation of $\mathbf{F}_k'$. We see then that the product $G_{kj}'G_{kj}$ has virtually the same distribution for any method of input-generation. In particular, we have that $G_{kj}'G_{kj} \sim \mathcal{N}(r, Mn_I)$. We conclude that the various methods of generating the input-vector are virtually equivalent from the viewpoint of the associator. From this point on, these methods will be discussed interchangeably.[7]

From this, we have also that the input-information provided by the coin method represents the maximum amount of information utilized by the associator for any mode of input-generation. This can be seen by replacing $p_F - 1/2$ by the equivalent $r/(2n_I)$ in equation (5.7) to get

$$I(\mathbf{F}_k' : \mathbf{F}_k) \approx \frac{(\log_2 e)r^2}{2n_I} \tag{5.13}$$

This information is less than r bits when $r < n_I/\log_2 e$. This will be the case in the analysis to follow since (5.13) is necessary for (5.7) to hold. We conclude that the coin method provides the smallest input-information compared with the other methods (the neighborhood method provides roughly the same amount of input-information as the coin method). Because the associator sees no difference in these methods, the input-information provided by the coin method must be the maximum amount useful to the associator when computing the output vector. The coin method of generation can therefore be used to ascertain the performance of the associator despite of the actual method of input-generation. This allows us to exploit the simplicity of analysis afforded by the coin method while retaining the generality to performance under the other input-generation modes.

We now begin to calculate the probability p_G that $G_{kj}'' = G_{kj}$ which is the same as the probability that $G_{kj}'G_{kj} \geq 0$. Since the product $G_{kj}'G_{kj}$ is normal with mean $(2p_F - 1)n_I$ and variance Mn_I, the probability p_G is easily determined

$$\begin{aligned} p_G &= P(G_{kj}'G_{kj} \geq 0) \\ &= 1 - P(G_{kj}'G_{kj} \leq 0) \end{aligned}$$

[7] The equivalence of the neighborhood method to the coin method follows from the fact that the vast majority of vectors in the interior of the ball in (5.5) lie near the boundary provided the radius is less than $n/2$ (see Kanerva [28]). The ball method and coin method will be consistent if the radius of the ball is roughly $n_I(1 - p_F)$ (see appendix B). The distribution of vectors generated via either method is that of a "ring" surrounding the central category-prototype. The "thickness" of the ring being determined by the variance of the coin method.

$$= 1 - Pr(G_{kj}'G_{kj} \text{ is } (2p_F - 1)n_I/\sqrt{Mn_I} \text{ standard dev's below the mean})$$

$$= 1 - \Phi\left(\frac{-(2p_F - 1)n_F}{\sqrt{Mn_I}}\right)$$

$$= \Phi((2p_F - 1)\sqrt{n_I/M}) \qquad \text{since } \Phi(x) = 1 - \Phi(-x) \tag{5.14}$$

where Φ is the standard normal distribution function. Since $p_G < 1$, and M will generally be larger than n_I, it follows that $(2p_F - 1)\sqrt{n_I/M}$ is typically less than 1. This allows use of the Taylor approximation to Φ given in chapter 2 page 19. We get

$$p_G \approx \frac{1}{2} + \frac{1}{\sqrt{2\pi}}\sqrt{n_I/M}(2p_F - 1) = \frac{1}{2} + \sqrt{2n_I/\pi M}(p_F - 1/2) \tag{5.15}$$

In a manner similar to the derivation of equation (5.7) we have

$$I(\mathbf{G}_k'' : \mathbf{G}_k) \geq n_O(1 - \mathcal{H}(p_G))$$

$$\approx 2n_O(\log_2 e)(p_G - \frac{1}{2})^2 , \qquad 0.5 \leq p_G \leq 0.88 \tag{5.16}$$

Assuming p_G is in the stated range, we appeal to (5.15) and substitute $\sqrt{2n_I/\pi M}(p_F - 1/2)$ for $p_G - 1/2$ in (5.16)

$$I(\mathbf{G}_k'' : \mathbf{G}_k) \geq 2n_O(\log_2 e)\frac{2n_I}{\pi M}(p_F - 1/2)^2$$

$$\approx \frac{2n_O}{\pi M} I(\mathbf{F}_k' : \mathbf{F}_k) \tag{5.17}$$

where the second approximation is due to (5.7). Dividing by $I(\mathbf{F}_k' : \mathbf{F}_k)$ (assumed larger than zero), we have a lower bound on the throughput of the associator

$$T(\mathbf{W}) \geq \frac{2n_O}{\pi M} \tag{5.18}$$

5.3. Classifiable Inputs

5.3.1. Lower Bounds on Input Information

As stated earlier, the redundancy, R, must be larger than $1/T(\mathbf{W})$ for reliable classification. Now that the throughput of the associator has been found, we have the lower bound

$$R \geq \frac{\pi M}{2n_O} \tag{5.19}$$

By definition (5.3), the input-information is given by

$$I(\mathbf{F}_k' : \mathbf{F}_k) = R\log_2 M \tag{5.20}$$

Together, (5.19) and (5.20) imply a lower bound on the input-information

$$I(\mathbf{F}_k' : \mathbf{F}_k) \geq \frac{\pi M\log_2 M}{2n_O} \tag{5.21}$$

By our assumption, $\mathbf{F}_k'$ is generated by the coin method. Thus the **bitwise information** $I(F_{ki}' : F_{ki})$, $i = 1, 2, \ldots, M$ is independent of $i = \{1, 2, \ldots, M\}$. Also the input-information $I(\mathbf{F}_k' : \mathbf{F}_k)$ is given by (5.6). We conclude that the input-information is n_I times the bitwise information. Dividing relation (5.21) by n_I, we get the lower bound

$$I(F_{ki}' : F_{ki}) \geq \frac{\pi M\log_2 M}{2N} \tag{5.22}$$

for the bit-wise information.

5.3.2. Lower Bounds on Feature Size

We can obtain minimal requirements on p_F and r by inverting the approximations of (5.7) and (5.13) to get each parameter in terms of $I(\mathbf{F}_k' : \mathbf{F}_k)$. From (5.7) and the assumption that $p_F > 1/2$ we have

$$p_F = \frac{1}{2} + \sqrt{I(\mathbf{F}_k' : \mathbf{F}_k)/(2n_I\log_2 e)}$$

$$= \frac{1}{2}\left(1 + \sqrt{2I(\mathbf{F}_k' : \mathbf{F}_k)/(n_I\log_2 e)}\right) \tag{5.23}$$

The relation for r is obtained from (5.13), (5.23) and the fact that $r = (2p_F - 1)n_I$

$$r \approx \sqrt{2n_I I(\mathbf{F}_k' : \mathbf{F}_k)/\log_2 e} \tag{5.24}$$

where $I(\mathbf{F}_k' : \mathbf{F}_k)/\log_2 e$ is the input-information in natural-logarithm units or "nats". Using equation (5.20) we get p_F in terms of the redundancy

$$p_F = \frac{1}{2} + \sqrt{R \ln M/(2n_I)} \tag{5.25}$$

Similarly for r,

$$r \approx \sqrt{2n_I R \ln M} \tag{5.26}$$

The lower bound (5.19) for R gives a lower bound for each parameter

$$p_F \geq \frac{1}{2}(1 + \sqrt{\pi M \ln M/N}) \tag{5.27}$$

and

$$r \geq \sqrt{(n_I/n_O)\cdot \pi M \ln M} \tag{5.28}$$

This means that if $\mathbf{F}_k'$ is generated from $\mathbf{F}_k$ by copying r of $\mathbf{F}_k$'s components we need to copy at least $\lceil \sqrt{(n_I/n_O)\pi M \ln M} \rceil$ components for classification to be possible. Reliable classification requires that this number be the minimum feature-size allowable for the input-vector if it is a single-feature vector. The number of non-overlapping features (sub-vectors) an input-vector can have is obviously the dimensionality of the vector divided by the minimal feature-size $\lfloor n_I/\lceil \sqrt{(n_I/n_O)\pi M \ln M} \rceil \rfloor$. If we let f_{min} be the minimal feature size and n_{max} be the maximal number of non-overlapping features allowable in an input-vector, then we have roughly

$$f_{min} \approx \sqrt{(n_I/n_O)\pi M \ln M} \tag{5.29}$$

and

$$n_{max} \approx \sqrt{N/(\pi M \ln M)} \tag{5.30}$$

As shown later, the fraction under the radical in (5.30) cannot be less than one for reliable classification. We see then that if we are to have n non-overlapping features in our vectors, then the number of weights in the associator will have to exceed $\pi M \ln M$ by a factor of n^2. This is a rather heavy price to pay for the ability to classify vectors on the basis of a single feature.

We make one important observation regarding the information content of an n-dimensional ± 1-vector. If X is the number of 1's that occur in a balanced-Bernoulli vector, $\mathbf{A}$, then X is a r.v. with mean $n/2$ and standard deviation $\sqrt{n}/2$. It stands to reason therefore, that a sub-vector of $\mathbf{A}$ of length $\sqrt{n}/2$ represents a unit of information of $\mathbf{A}$. To verify this, let R be the redundancy (as defined by (5.3) for some $M > 0$) of the information that $\mathbf{A}$ is to provide about another vector, $\mathbf{B}$. If we are to copy components of $\mathbf{B}$ to $\mathbf{A}$, then equation (5.26) gives the minimal number r of components that should be copied (the rest are chosen independently of the components of $\mathbf{B}$). This number can be expressed in terms of the number of standard-deviation-length sub-vectors needed

$$r = 2\sqrt{2R \ln M}\cdot(\sqrt{n}/2) \tag{5.31}$$

To provide $R \log_2 M$ bits of information, we must copy at least $2\sqrt{2R \ln M}$ sub-vector "units" of information from $\mathbf{B}$. The "square-root" relationship between the number of bits of information and the number of sub-vector "units" is due to the quadratic dependence of information on the probability that a component of one vector agrees with its counterpart in another vector (see relation (5.7)). The fact that information in balanced-Bernoulli vectors is closely related to $\sqrt{n_f}/2$-length sub-vectors must play a part of any mode of representation that codes information into ± 1-vectors. If information coded into sub-regions of the input-vector is to provide the sole cue to an associator for classification, the subregions must cover at least $2\sqrt{2R \ln M}$ sub-vector "units" of the input-vector, where R is the minimal input-redundancy required by the associator.

5.3.3. Fraction of the Input Space that is Classifiable

An analysis of minimal requirements for the neighborhood method of input-generation are derived in appendix B. Because this method is roughly equivalent to the coin method and because it gives us an estimate of the number of vectors that can be classified, we relate the results here. First, for a ball centered about an input-prototype, if a randomly chosen vector from the ball is to provide $R \log_2 M$ bits of information about the prototype, then the ball must comprise M^{-R} of the input-space. From appendix B, the radius ρ is roughly

$$\rho \approx \frac{n_I}{2} - \frac{\sqrt{n_I}}{2}\sqrt{2R\ln M - \ln(4\pi R\ln M)} \tag{5.32}$$

The lower bound on the redundancy in (5.19) gives an upper bound on the radius

$$\rho \leq \frac{n_I}{2} - \frac{\sqrt{n_I}}{2}\sqrt{\pi M\ln M/n_O - \ln(2\pi^2 M\ln M)/n_O} \tag{5.33}$$

In appendix B, geometrical considerations of the output space suggest that this radius is too large. The excess redundancy required however should not be more than twice the minimum (see appendix B for a discussion of this point). This gives us a lower bound for ρ

$$\rho \geq \frac{n_I}{2} - \frac{\sqrt{n_I}}{2}\sqrt{2\pi M\ln M/n_O - \ln(4\pi^2 M\ln M)/n_O} \tag{5.34}$$

We now derive the upper bound on the fraction of the input-space that can be classified. This result is obtained from the lower bound on the information required at the associator input. Since the associator produces an output on the basis of the Hamming-distance between the input-vector and the input-prototypes, input-vectors providing the associator a specified amount of information about an input-prototype should come from a set of vectors nearest to the prototype. If the set is a ball of radius ρ about the prototype, then random selection of a vector from the ball (neighborhood method of input-generation) is roughly equivalent to the coin method of input-generation when $\rho \approx n_I(1 - p_F)$. When an input-vector $\mathbf{F}_k'$ is generated by the neighborhood method, and the information it provides about $\mathbf{F}_k$ is $I(\mathbf{F}_k' : \mathbf{F}_k)$, the ball it comes from will encompass $\exp_2(-I(\mathbf{F}_k' : \mathbf{F}_k))$ of the total input-space. For our system, there are M balls surrounding M input-prototypes so the total fraction of the input space covered by the M balls is at most $M\cdot\exp_2(-I(\mathbf{F}_k' : \mathbf{F}_k))$. The regions could overlap, though the overlap will be negligible if the input-information is at least $2\log_2 M$. Now if R is the redundancy of the input, then the input information is $R\log_2 M$ bits and the fraction C of the input-space that is classifiable is

$$C \approx M^{1-R} \tag{5.35}$$

Using the lower bound on R we have the upper bound on C. In fact, as we shall see later, M will usually be greater than n_O by a large factor so that the fraction of the space that is classifiable will be quite small.

$$C \leq M^{1-\pi M/(2n_O)} \tag{5.36}$$

where M is assumed to be larger than n_O.

5.3.4. Restrictions on Matrix Dimensions

The inequality of (5.33) is *required* for reliable classification, whereas inequality (5.34) is merely a reasonable bound on how small the value of ρ need be made to insure the system will work. Therefore inequality (5.33) must be larger than zero if the system is to classify its inputs. This constraint leads to a lower bound on N which will be derived by different means later (see equation (5.42)). The lower bound on N is the minimal number of vectors required merely for *storing* the prototypes when the Hopfield non-linearity is present at the associator output.

An even tighter constraint on the required matrix size is obtained when we require that the system be capable of classifying "highly-degraded" input-vectors. A highly-degraded input-vector is a vector that is nearly orthogonal to its category-exemplar (the nearest input-prototype). From (5.33), we see that classification of such inputs is possible when n_I is large compared to $\sqrt{\pi M \ln M}\sqrt{n_I/n_O}$. In this case, if ρ is near the theoretical maximum given in (5.33), the input-vectors at the edge of the neighborhood of a prototype will be at a Hamming-distance nearly $n_I/2$ from the prototype. A reasonable way to make n_I large enough is to require $n_I \geq 8\sqrt{\pi M \ln M}\sqrt{n_I/n_O}$. Multiplying through by $\sqrt{n_O/n_I}$ and squaring both sides of this inequality gives us a lower bound on the number N of weights

$$N \geq 64\pi M \ln M \tag{5.37}$$

Comparing this to the requirement (5.42) for storage, we see that classification of "highly-degraded" input-vectors requires roughly 50-100 times the number of weights required for merely storing the prototype vectors.

We note a few restrictions on the parameters inferred by the analysis in appendix B. First, if the input-vector is to have a redundancy no greater than R (keeping R low, makes a larger portion of the input-space classifiable, see equation (5.35)), then we must have $\rho > 0$ in equation (B.6), page 109. This becomes the constraint

$$n_I \geq 2R \ln M \tag{5.38}$$

This constraint applies equally well for the output dimensionality with R between 1 and 2 so that

$$n_O \geq 2\ln M \tag{5.39}$$

is a minimal requirement for the output-dimensionality (see equation (B.8)). In the "throughput" section, restrictions on the parameters n_I, n_O and M were also made to obtain the approximations used to obtain the associator throughput. The linear approximation made in equation (5.14) assumed that M was at least as large as n_I. This assumption assures that the argument to Φ was no larger than 1 so that higher terms in the Taylor approximation to Φ can be dropped.

The assumption that the argument to Φ in equation (5.14) was less than 1 leads to a restriction on p_G. This assumption together with (5.15) gives the upper bound

$$p_G \leq \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \approx 0.9 \tag{5.40}$$

These relations illustrate the limitations of the theory that has been developed. A designer of an associator on M associations must stay within the parameter-assumptions in order for the performance predictions of the theory to apply.

5.4. Performance Degradation Due to Non-Linear Output

The "Hopfield non-linearity" in figure 5-3 was introduced for the sake of simplifying the analysis. The problem of determining the information $I(\mathbf{G}_k'; \mathbf{G}_k)$ available directly from the associator-matrix is somewhat more difficult than finding the information $I(\mathbf{G}_k''; \mathbf{G}_k)$ available from the non-linearity. Unfortunately, however, addition of the non-linearity eliminates much of the information available from $\mathbf{G}_k'$. That this is so is evidenced by the degradation of storage capacity due to the non-linearity.

To estimate the storage capacity of the non-linear associator in figure 5-3, put $p_F = 1$ to constrain the input vectors to belong to the set of input-prototypes. The formula p_G that gives p_G in terms of p_F becomes

$$p_G \approx \frac{1}{2}(1 + \sqrt{2n_I/\pi M}) \tag{5.41}$$

This approximation is good when p_G is near 1/2, so in particular, M must be at least n_I an (5.41). The approximation was obtained from (5.15) which is a linearization of the normal distribution function $\Phi(x)$ about $x = 0$. It overestimates p_G with the overestimate becoming large as p_G nears 1. In fact one pays a high penalty in storage capacity when insisting that each bit of $\mathbf{G}_k''$ match its counterpart in

$\mathbf{G}_k$ with high probability. This is due seen from the fact that when n_I M is increased p_G does not increase as rapidly as (5.15) would indicate. In any event, using equation (5.15) will give an upper bound on the storage capacity.

As stated in the chapter on storage capacity, useful storage requires the output information to be at least $\log_2 M$ bits. During retrieval, the number of bits present at the input is n_I. If we multiply n_I by the throughput $T(\mathbf{W})$ and require the result to be larger than $\log_2 M$, a constraint on the matrix size is obtained. Unfortuneately $T(\mathbf{W})$ was obtained by assuming p_F was not too near 1. We will have to use equations (5.41) and (5.16) instead to get the constraint. Remember however, (5.41) assumes p_G is not too near 1, which will be the case if $M \geq 2n_I$. From (5.41) and (5.16) we have

$$I(\mathbf{G}_k'' : \mathbf{G}_k) \approx \frac{M \log_2 e}{\pi M}$$

By the constraint (5.2), the right-hand-side must be larger than $\log_2 M$. The resulting inequality can then be rearranged to get

$$\frac{\pi M \ln M}{N} \leq 1 \tag{5.42}$$

To put (5.42) another way, N must be at least $O(M \ln M)$. This is a stronger requirement than the one derived for storage in the previous chapter. This new bound implies that if n_O is $O(\ln M)$, then n_I must be $O(M)$.

If errors are allowed at the output of the second stage of figure 5-3 then the storage can be increased. If P_e is the error probability, then for $0 < P_e \leq 1/2$, M large, we need $(1 - P_e)\log_2 M$ bits at the output. From this and (5.16) we have

$$2n_O(\log_2 e)(p_G - 1/2)^2 \geq (1 - P_e)\log_2 M \tag{5.43}$$

and from (5.41)

$$2n_O(\log_2 e)\frac{n_I}{2\pi M} \geq (1 - P_e)\log_2 M$$

which gives

$$\frac{\pi M \ln M}{N} \geq \frac{1}{1 - P_e} \tag{5.44}$$

As with the case with storage treated in the previous chapter, the number of required weights is proportional to $1 - P_e$. On the other hand, the maximal value of M no longer increases in proportion to $1/(1 - P_e)$.

The reason the non-linearity decreases the information content of the output of the associator is that it forces the best-match process of figure 5-3 to "count" the number of places that the output $\mathbf{G}_k'$ disagrees in sign with $\mathbf{G}_k$ (recall the method of computing $\mathbf{G}_k''$). This can be seen from figure 5-3 with the non-linearity removed and from equation (5.12) which is the formula for one summand-term in the dot-product $\mathbf{G}_k' \cdot \mathbf{G}_k$. If the best-match process in figure 5-3 uses the output of the associator-matrix directly, it can use the dot-product similarity-measure to compare $\mathbf{G}_k'$ with every one of the output-prototypes. Now, a single summand in the dot-product $\sum_j G_{kj}' \cdot G_{kj}$ is binomially distributed with positive mean $(2p_F - 1)n_I$. Such a term will tend to have larger magnitude when it is positive than when it is negative. This means that the dot-product can do more than "count" how many positions G_{kj}' agree in sign with their counterparts G_{kj}. The dot-product also uses "magnitude" information to ascertain the "confidence" that a specific component of $\mathbf{G}_k'$ is of the proper sign. On the other hand, whether the performance limits of the previous chapter can be achieved depends on whether retrieval in the linear-associator is optimal. For this to be so, the full entropy of the matrix (per storage item) must be available at the memory output. What's more, the information available must be *useful* to the best-match process.

The analysis of the linear case should entail evaluation of the information content of $\mathbf{G}_k'$ by evaluating it as a rendition of the "signal" $\mathbf{G}_k$ with added binomial "noise". The "signal-to-noise ratio" as a function of M would then be used to quantify the information content. The analysis is similar in concept with evaluation of information contained by a gaussian signal in the presence of gaussian noise (see Gallager, [12, p. 32, Example 4]). The difference is that the "signal" components G_{kj} are not gaussian but Bernoulli r.v.'s and the "noise" in $\mathbf{G}_k'$ due to the associator-matrix is binomial rather than gaussian. These differences are responsible for the difficulty in determining the information $I(\mathbf{G}_k'; \mathbf{G}_k)$. The difficulties are not insurmountable, but the analysis may be as involved as that in Appendix A, since the problem of approximating a discrete entropy with a continuous one in the appendix seems related to the problem of approximating the information in $\mathbf{G}_k'$.

5.5. Classifier Design Considerations

At this point,we are ready to illustrate the design of an associator to meet specific requirements. Two designs will be given to show how the relative sizes of parameters interact. Given the number M of categories, a fraction α of the space to be classified and the maximum classification error-probability,

P_e, we wish to find the dimensions n_I and n_O that result in a matrix of minimal size N that meet the requirements.[8] To begin, let $P_e = 0$ for simplicity. Notice that a ball $B_k(\rho)$ about a prototype must contain about α/M of the input space. Since the fraction of input-vectors in the ball is $\exp_2(-I(\mathbf{F}_k'; \mathbf{F}_k))$, we have

$$\frac{\alpha}{M} = \exp_2(-I(\mathbf{F}_k'; \mathbf{F}_k)) \tag{5.45}$$

so that

$$I(\mathbf{F}_k'; \mathbf{F}_k) = \log_2 M - \log_2 \alpha \tag{5.46}$$

Now $R = I(\mathbf{F}_k'; \mathbf{F}_k)/\log_2 M$ so by (5.46) we have $\log_2 \alpha - \log_2 M = -R\log_2 M$. Rearranging and converting to natural logarithms gives a more convenient form

$$R = 1 + \frac{-\ln \alpha}{\ln M} \tag{5.47}$$

The two classifiers we produce will be called the **large-α model** and the **small-α model**.[9] The large-α model will have $-\ln \alpha$ proportional to $\ln M$, so that for some positive $K \geq 10$ we write

$$-\ln \alpha = K \ln M \tag{5.48}$$

The small-α model assumes that $-\ln \alpha$ is proportional to M. In this case we put

$$-\ln \alpha = \frac{M}{K} \tag{5.49}$$

with $K \leq M/(10 \ln M)$. Calculating the redundancy from (5.47) for the large-α model we have

$$R = 1 + K \approx K \tag{5.50}$$

and for the small-α model

---

[8] Of course, a design problem may differ as to which parameters are initially specified. Most notably is the case when a designer is dealing with an input-space whose vector-dimensionality n_I is already known.

[9] Since $0 < \alpha < 1$, the quantity $-\ln \alpha$ is positive and grows without bound as $\alpha \to 0$. The terms "large-α" and "small-α" are of course relative. A large-α model will only classify a small portion of the input-space. A small-α model will classify a portion orders of magnitude smaller. Even in the case of the small-α model however, there are $\exp_2(n_I)$ possible input-vectors so that the actual *number* of vectors classifiable is still very large.

$$R = 1 - \frac{M}{K \ln M} \approx \frac{M}{K \ln M} \tag{5.51}$$

Recall that relation (5.19) must hold for reliable classification. From this we get the lower bound on n_O

$$n_O \geq \frac{\pi M}{2R} \tag{5.52}$$

For the large-α model, this implies

$$n_O \geq \pi M/(2K) \tag{5.53}$$

For the small-α model

$$n_O = \frac{\pi}{2} K \ln M \tag{5.54}$$

To get a constraint on n_I, we use the fact that the maximum Hamming-distance between an input-vector an its category-exemplar is *roughly*

$$\rho_{max} \approx \frac{n_I}{2} - \frac{\sqrt{n_I}}{2} \cdot \sqrt{2R \ln M} \tag{5.55}$$

If we are to classify vectors that are nearly orthogonal to their category vectors, *then* ρ_{max} should be nearly $n_I/2$. For the large-α model, this is more important than for the small-α model since the former must classify more of its input-space. The closer ρ_{max} is to $n_I/2$ however, the more weights are required for either model given a fixed value of K. For the sake of comparison then, we will use the same value $\rho \approx (2/3)n_I/2$ for both models. This isn't much of a constraint. A better one is $\rho_{max} = (9/10)n_I/2$ but *the number of* weights required would be about 10 times as large. From equation (5.55) and our constraint, we get

$$\sqrt{n_I} \approx 3\sqrt{2R \ln M}$$

so that

$$n_I = 18R \ln M \approx 20R \ln M \tag{5.56}$$

For the large-α model $R \approx K$ so

$$n_I = 20K \ln M \tag{5.57}$$

whereas the small-α model has $R = M/(K \ln M)$ so

$$n_I = \frac{20M}{K} \tag{5.58}$$

The number N of weights in both cases is $10\pi M \ln M$ or 10 times the minimum required for storing M prototypes.

The thing to notice is that the large-α model has n_I of order $\ln M$ and n_O of order M. In other words, the input-dimensionality far exceeds the input-dimensionality. In order to classify such a large portion of the input-space, the input-redundancy must not be large. This is seen from relation (5.47). When $\alpha \to 1$, we have $-\ln \alpha \to 0$ so that $R \to 1$. The throughput of the system must be large so many units are needed to produce the output.

For the small-α the situation is reversed. The input-dimensionality is large and so can accomodate the large input-redundancy (The redundancy can never exceed $n_I/\log_2 M$). The number of units can be small since the high redundancy insures adequate output information even with low throughput.

As a numerical example, suppose that $M = 50{,}000$ and to assure $M \geq n_I$ in (5.58), let $K = 50$. For the large-α model, $R = 50$ so by (5.57) $n_I \approx 10{,}800$, and by (5.53) $n_O \approx 1570$. For the small-α model $R = 92$, equation (5.58) implies $n_I = 20{,}000$ and (5.54) gives $n_O = 850$. Both models have roughly $1.7 \cdot 10^7$ weights.

Now let ς be the number of classifiable vectors in each case. We want to estimate the entropy $\log_2 \varsigma$ of the classifiable portion of the input-space. By equation (5.35), this entropy is roughly $\log_2 (M^{1-R} \exp_2(n_I))$, or approximately $\varsigma = n_I + (1 - R)\log_2 M$. By equation (5.47) we have

$$\varsigma = n_I + \log_2 \alpha \tag{5.59}$$

For the large-α model, $\varsigma = n_I - K \log_2 M \approx 10{,}000$. For the small-$\alpha$ model, $\varsigma = n_I - M \log_2 e / K \approx 18{,}600$. The proportion of the space classified by the large-α model is 10^{-240} whereas the small-α model classifies roughly 10^{-440} of its input-space (computed from the respective values of α).

The moral however, is that the small-α does not classify fewer vectors than does the large-α model. The input-space for the small-α model is so much larger than that of the large-α model that the actual *number* of vectors classifiable by the small-α is much larger. In fact, the number of vectors that can be classified by the small-α model dwarfs the number of vectors in the entire input-space of the large-α model.

One way of viewing this numerical advantage of the small-α model is in terms of **data compression**. Whereas the number of input-vectors to be classified is potentially very large, the number M of categories at the output is relatively miniscule (the number of categories should be less than the number of weights or even smaller). The entropy of the output relative to that of the input is therefore quite small and this is what is meant by "data-compression". The fact that the matrix faces less information at its output than at its input should be reflected by its architecture if high-performance is expected. For a classifier with N weights that is to classify a large number of input-vectors, the output-dimensionality should be as small as possible (within the constraints described in appendix B) compared with the input-dimensionality. Such a system will classify a maximal number of input-vectors for a given number of associations (categories) stored.

One should also notice that the classifier classifies only a very small portion of the input-space. This results in a "*double-data-compression*". Most inputs are simply not considered to be valid input "signals". Those that are will then be mapped to a relatively small number of categories. The final result is an output that has far less entropy than the total input-space. We conclude that the associator-as-classifier assumes that most of the space of possible inputs are irrelevant to its task. The portion of the space that is considered relevant is specified by the collection of prototype-vectors. These in turn specify the pertinent informational-features of the input-space. All other information is ignored, resulting in an output that is a compact representation of the salient features of the input.

5.6. Maximal Performance and Figures of Merit

5.6.1. Merit Parameters and Figures of Merit

We define a **merit-parameter** to be some measure of system performance with regard to storage or classification. In the case that there is a maximal value for the merit-parameter, we divide the merit-parameter by the maximal value to get a **figure-of-merit**. The maximal value for the parameter is determined via information-theoretic constraints on an arbitrary memory/classification system and so is independent of features specific to a particular device. The figure-of-merit will generally take on a value between zero and one with the value "1" corresponding to optimal performance. Thus the merit-figure can be used for comparison of various systems whose merit-figures are known.

5.6.2. Load, Efficiency, Throughput and Retrievable Information

In the chapter on storage, we derived a figure-of-merit L called the load. It was defined as the ratio of the number of items stored (a merit-parameter) divided by the number of items storable. Another figure-of-merit we defined was called the efficiency, η, that was the ratio of the number of bits stored in the memory divided by the number of bits required to represent the *memory* itself. For classification, it is also desirable to obtain *relevant* merit-parameters and figures-of-merit.

An obvious merit-parameter for classification is the throughput $T(\mathbf{W})$ defined earlier. The optimal value T_o can be be derived for an arbitrary memory obeying relation (3.13). The **throughput-merit**, τ, of a system is then defined as $T(\mathbf{W})/T_o$. To obtain T_o, we divide the maximum-possible output-information by the minimum allowable input-information. For systems obeying equation (3.13), the maximum output-information per association is $H(\mathbf{W}, M)/M$. The minimal input-information required is $\log_2 M$ bits so we have

$$T_o = H(\mathbf{W}, M)/(M\log_2 M) \tag{5.60}$$

So the throughput-merit is given by

$$\tau \equiv \frac{T(\mathbf{W})}{T_o} = \frac{T(\mathbf{W})M\log_2 M}{H(\mathbf{W}, M)}$$

where

$$\frac{M\log_2 M}{H(\mathbf{W}, M)} \leq \frac{M \cdot I(\mathbf{G}_k'' ; \mathbf{G}_k)}{H(\mathbf{W}, M)} \leq 1 \tag{5.61}$$

If we use the fact that $H(\mathbf{W}, M) \approx (1/2)M\log_2 M$ then the figure-of-merit τ for linear-associator systems satisfies

$$\tau = \frac{T(\mathbf{W})M\log_2 M}{(1/2)M\log_2 M} = T(\mathbf{W})\frac{2M}{N} = T(\mathbf{W}) \cdot L \tag{5.62}$$

where L is the load. Thus the throughput-merit for the outer-product associator is just the product of the two merit parameters derived earlier. This product however has the additional property that it can never exceed 1. It would be of interest as to whether the throughput-merit for the linear-associator (without the Hopfield non-linearity) is roughly equal to 1 (or at least constant) for a large range of values of the load. If so, we'd have that the throughput trades directly with load as more associations are stored.

In any event, we have that

$$T(\mathbf{W}) \leq \frac{1}{L} = \frac{N}{2M} \tag{5.63}$$

for the linear-associator. For the associator with no non-linearity then, the upper bound can be quite large when M is much smaller than N.

For the case that the Hopfield non-linearity is present at the matrix-output, we can obtain the maximum r achievable by the associator (see figure 5-3). By (5.42), the number N in (5.62) is larger than $\pi M \ln M$. Replacing N by this value in (5.62) gives the upper bound

$$r \leq T(\mathbf{W}) \frac{2M}{\pi M \ln M} = \frac{2T(\mathbf{W})}{\pi \ln M} \tag{5.64}$$

Since $T(\mathbf{W}) = 2n_O/(\pi M)$, we have the bound

$$r \leq \frac{2n_O}{\pi M} \frac{2}{\pi \ln M} = \frac{4n_O}{\pi^2 M \ln M} \tag{5.65}$$

which is much smaller than 1 if the number of stored prototypes is larger than n_O. By way of comparison, the *linear* associator could conceivably have a r as large as 1. However this has not been established since the throughput of the linear-associator has not been determined.

A figure-of-merit relevant to the memory is apparent from the results of chapter 2. By relation (3.13), we have $I(\mathbf{G}_k'' : \mathbf{G}_k) \leq H(\mathbf{W})$. Therefore the **retrievable-fraction** *of stored information* is

$$\mu \equiv \frac{M I(\mathbf{G}_k'' : \mathbf{G}_k)}{H(\mathbf{W})} \tag{5.66}$$

The retrievable fraction, by relation (3.14), cannot exceed 1.

For the non-linear associator, we can find the maximal retrievable-fraction from knowledge of the throughput. Remembering that the largest that the input-information can be is n_I bits, we use the definition of throughput to get

$$\mu = \frac{M T(\mathbf{W}) I(\mathbf{F}_k' : \mathbf{F}_k)}{(1/2) M \log M} \leq \frac{M (2n_O/\pi M) n_I}{(1/2) M \log_2 M} = \frac{4}{\pi \log_2 M} \tag{5.67}$$

This parameter is quite small for large systems that store many associations. For the Hopfield-non-linear associator, systems become extremely sub-optimal as the system-size gets large.

5.6.3. Search for an Overall Figure of Merit for Memory

It would be preferable if an overall figure-of-merit for memory-performance could be found. This figure, called the **memory-merit**, $\mathcal{M}$, should reflect all aspects of memory operation and have the property that a memory could in principle attain a memory-merit of one. An attempt to define $\mathcal{M}$ might involve taking the product of τ, μ, and η to get

$$\mathcal{M} = \tau\mu\eta \tag{5.68}$$

For memory systems whose load L can be defined, one can restrict consideration to memories that are not overloaded (i.e. $L \leq 1$). The load could then be incorporated into $\mathcal{M}$

$$\mathcal{M} = \tau\mu\eta L \tag{5.69}$$

The efficiency η is just related to the representation used for the weights of the memory and is therefore indicative of limitations of the memory's implementation. This parameter should be dropped if only the memory's inherent properties are to be considered

$$\mathcal{M} = \tau\mu L \tag{5.70}$$

If there is a general figure-of-merit for memory, this last one may be close to the mark. On the other hand, we saw in relations (5.61), (5.62) and (5.66) that τ is related to both μ and L, so one may wonder if $\mathcal{M}$ in (5.70) may contain redundant information. Also, there may be tradeoffs that force the value of one of the factors in (5.70) to be low when the other is high. If this true even in principle, then it is possible that no memory can achieve a merit of one and the memory-merit would not satisfy the definition of a figure-of-merit. This possibility seems unlikely based on calculations done by the author. In fact, if the outer-product linear-associator has an optimal throughput (τ near one for large systems), it is possible that it could be have a memory-merit approaching one as the associator size gets large.

5.6.4. Classification Figures of Merit

For classification, a merit parameter that can be "normalized" to produce a figure-of-merit is hard to obtain without imposing artificial constraints. One merit measure worthy of consideration however is the ratio of the bits needed to encode the classifiable input set to the number of bits needed to represent the categories at the output. This is called the **fan-in**. The parameter is of interest because it represents

the capability of the system to react to a very large input-space when it has stored a relatively small "representation-space". Indeed, this is the very essence of classification. A classifier "filters out" non-essential information allowing subsequent systems to provide for far fewer contingencies. Unfortunately, a classifier can achieve a high fan-in by classifying all possible input-vectors into one "category".

One remedy, is to multiply the fan-in by the storage-load of the system. A system with a large load will have stored a maximal number of categories and so the product of the fan-in and load will be maximized by systems that can classify a large portion of the input-space even when storing a large number M of categories. With this in mind, we consider the fan-in alone when the number of categories is a fixed value M. We will derive the optimal of fan-in for this number of categories and use it to find the "normalized" **fan-in merit**.

To calculate the fan-in merit f_m for the linear-associator, note that the logarithm of the classifiable space is roughly $n_I + (1 - R)\log_2 M$ by equation (5.36), where R is the redundancy. The number of bits needed to label the M different categories is $\log_2 M$ bits so the fan-in f is

$$f = \frac{n_I + (1 - R)\log_2 M}{\log_2 M} = \frac{n_I}{\log_2 M} + 1 - R$$

where R is the input redundancy. Note that $n_I/\log_2 M$ is the maximum redundancy that can be facilitated by the input. To get a normalized figure of merit, we first make the constraint that the input-space has entropy n_I and the output-space being composed of M categories, has entropy $\log_2 M$. Also note that $R \geq 1/T(\mathbf{W}) \geq 1/T_o$, so by (5.60)

$$R \geq \frac{M\log_2 M}{H(\mathbf{W})} \tag{5.71}$$

and so the largest value f_o of f is defined by

$$f_o \equiv \frac{n_I}{\log_2 M} - 1 - \frac{M\log_2 M}{H(\mathbf{W})} \tag{5.72}$$

The fan-in merit f_m is then

$$f_m = f f_o \tag{5.73}$$

To get the merit for the non-linear associator, recall from relation (5.42) that $N \geq -M\ln M$ so that

$M\log_2 M / H(\mathbf{W}) \leq 2/(\pi \ln M)$ and because $R \geq \pi M/(2n_I)$ we have

$$f_m \leq \frac{n_I}{\log_2 M} + 1 - \frac{\pi M}{2n_O} \Big/ \frac{n_I}{\log_2 M} + 1 - \frac{2}{\pi \ln M} \tag{5.74}$$

One final consideration is a parameter that measures the ratio of the size of the classification space $\mathcal{C}$ to the size of the input space $\mathcal{F}$. The higher the ratio, the more of the input is classifiable. The ratio will be called the **inclusion** I and is defined by

$$I = \frac{|\mathcal{C}|}{|\mathcal{F}|} \tag{5.75}$$

The theoretical maximum for this ratio is M^{1-R} where R equals the lower bound in (5.71), so

$$I \leq M^{1 - M\log_2 M/H(\mathbf{W})} \tag{5.76}$$

So the **inclusion-merit** ι is I divided by this theoretical maximum. The result is

$$\iota = \frac{I}{M^{1 - M\log_2 M/H(\mathbf{W})}} \tag{5.77}$$

From previous considerations, the ι for the non-linear associator has the upper bound

$$\iota \leq M^{1 - \pi M/(2n_O)} / M^{1 - 2/\pi \ln M} = M^{2/(\pi \ln M) - \pi M/(2n_O)} \tag{5.78}$$

A good overall merit parameter for classification might be the product of the load, the fan-in merit, and the inclusion merit. The issue of finding an overall figure-of-merit for memory and classification might not be hard to address. The author has only recently defined these merit measures and has not yet fully explored the alternatives.

In passing, we might add that these figures of merit can be quantified for the linear-associator once the throughput of the linear version of the classifier can be determined. We conjecture that the linear-associator may be very nearly optimal in most respects when the matrix size is large. As far as non-linearities are concerned, any non-linearity will cause performance degradation. However, "sigmoid" non-linearities used in so many connectionist systems (see 22, 24, 40), will perform reasonably well if they are not too "steep". In particular, if the rising portion of the sigmoid is broad enough to encompass most of the variance of the components of the matrix-output-vector, most of the matrix-output information will

be retained. Though the author has not made the attempt, a "maximal steepness" necessary for negligible information loss should be easily obtainable using something like the tails-lemma of appendix A. Here, one would use the sigmoid to limit the range of values that the components can assume as was done for the matrix-weights in the previous chapter to improve efficiency. In any event, the Hopfield non-linearity represents a sigmoid with "infinite steepness" and so provides the lower-bound on performance for sigmoid-non-linear outer-product associators.

Chapter 6
Summary

6.1. Contributions and Accomplishments

The most important contribution of this work is the characterization of memory and storage in terms of information theory. For memory, the primary accomplishment was evaluation of the matrix-entropy and the proof that it bounds the retrievable information. The bound was subsequently used to determine the amount of information stored as a function of matrix-size and number of associations stored. A criterion for minimal performance was obtained through the definition of channel memory. This criterion was then used to bound the number of items storable. We also dealt with the notion of retrieving information via separate "accesses" to the memory, one for each item stored. Though information obtained this way is not the same as that actually stored in the matrix, we find that the latter is an upper bound on the former.

Use of the concept of the matrix-channel allowed us to characterize and evaluate classification of the associator. For this, the fundamental concept defined was the matrix-throughput which is the ratio of the output information to the input information. The simple linear relation between the two for the associator with Hopfield non-linearity allowed us to quantify the fraction of the input-space that is classifiable and obtain minimal requirements on sub-features of incomplete-input vectors needed for their proper classification. We also noted requirements on the matrix-size as they relate to the task required. We found that an associator with Hopfield non-linearity, expected to classify inputs that are nearly orthogonal to their category-exemplars, requires 50-100 times as many weights as does one that merely stores its prototypes. The latter system is a "degenerate" classifier. It can properly "categorize" an input vector if that vector is an input-prototype. Such a system would not be very robust in its classification of input-vectors that have a significant number of "bits" in error. In any event, there is obviously a tradeoff between the number of categories over which the associator can divide the input-space and the fraction of the input-space that can be classified. The more category discrimination required of the system, the fewer vectors can be classified given a fixed matrix-dimensionalities.

We mention that in some sense, the associator is not really doing classification unless the output-dimensionality is very nearly equal to the logarithm of the number of categories stored. We were merely interested in conditions under which the associator would pass through information useful to a subsequent stage that is to determine the category to which the input to the associator belongs (see second-stage of 5-3). An associator could be said to classify its inputs if the outputs it produced were much nearer to the output-prototypes than the respective input-vectors were to their exemplar-prototypes. In the case of the Hopfield-non-linear associator, the average distance of the matrix-output from the "correct" output-prototype is $n_O(1 - p_G)$. We can decrease this distance by forcing p_G to be near one or by keeping n_O small. The first of these can only be done by storing less than n_I categories where n_I is the dimension of the input-vectors (see equation (5.15), page 59). The second option is fortunately in keeping with optimal performance of the classifier. In fact, we found earlier that a large input-dimensionality allows classification of a very large number of vectors for a given matrix-size and storage-load. This is probably the most important finding concerning associator-classification. A matrix that "fans in" so that its input-dimension is much larger than its output dimension will give the best classification performance for a fixed matrix-size and number of stored categories. Thus we have an architectural specification based on information theory. A classifier does data-compression so that the output-handles much less entropy than does the input and the matrix dimensionalities should reflect this fact for optimal performance.

After evaluation of the performance of the system, we obtained figures of merit for both memory and classification performance. These were "normalized" with respect to optimal information-theoretic performance limits and so serve as a basis of comparison of general memory/classifier systems. The associator with Hopfield non-linearity was shown to perform suboptimally, in fact, disappointingly so. On the "up side", the Hopfield-non-linear system provides a lower bound for performance of associators with "sigmoid" type non-linearities.

6.2. Limitations of this Investigation and Future Directions

The main limitation of this work was that it did not address the information content of the actual matrix-output (labelled $\mathbf{G}_k'$ in figure 5-3). The problems with the analysis are mentioned on page 67. Once this issue is addressed, one may be able to determine the optimal performance of any associator with sigmoid non-linearity on its output. What's more, the storage bound was merely an upper bound to performance. Knowledge of the amount of information present in the matrix-output would determine just how tight this bound is. We also assumed that the information at the output of the matrix is all useful to a second-stage process that must classify the output-vectors. This is not necessarily true but is probably a good assumption due to the fact that the associator maps similar inputs to similar outputs and the fact that we characterized information at both input and output in terms of vector-similarity.

A rather serious shortcoming of the analysis was that it assumed that the prototype vectors were chosen randomly, that is they were "balanced-Bernoulli" vectors. In reality, if a system acquires its prototypes by encoding representations of "stimuli" or "concepts" etc., it will most likely have correlated prototypes. So while we did not require orthogonality of the prototypes, the requirement that they be uncorrelated (randomly selected) is too stringent. The problem is confounded by the fact that storage capacity most probably degrades in the presence of inter-prototype correlation; the sensitivity to correlation becomes more pronounced as the system-size gets large.[10] This is a serious flaw since it indicates that the storage capacity may not be achievable in practice. On the other hand, the relation of mutual information to vector geometry outlined in appendix B may provide a means by which a set of prototypes can be strategically chosen so as to minimize correlation or equivalently maximize mutual Hamming-distance. If such a method could be easily incorporated into the encoding process, these systems could in fact achieve better-than-optimal performance since "de-correlation" could produce prototypes more mutually distant than random selection can.

Another issue not addressed was classification performance when the number of stored categories was less than the input-dimensionality. The analysis in the classification chapter would probably extend to this case if the linear approximation to Φ on page 50 was changed to a quadratic one for more accuracy. Even without this change however, the linear approximation overestimates p_G so the performance bounds derived in the classification chapter apply to the case that the number of stored categories is small. The upper bound merely becomes looser. As the number of stored categories is diminished, p_G increases but not as rapidly as the linear approximation would indicate. Note that even when the number of categories is less than the input-dimensionality, the analysis applies to *randomly selected* input-prototypes not orthogonalized (forcefully-decorrelated) prototypes. This is an advantage since it represents a relaxation of the orthogonality restriction needed for perfect retrieval (see [21, p. 18]).

Regarding future directions, there are too many possible avenues for continuing this work to mention here. Two however are of primary concern to the author. First is the analysis of the auto-associator as both memory and classifier. This extension is not without obstacles however. With respect to memory, the weights of an outer-product matrix are less independent when the output-prototypes are identical to the input-prototypes. On the other hand, the individual weights (excluding those on the diagonal which are constant and so contribute nothing to the matrix-entropy) will have the same distribution as those of the hetero-associator and should be nearly independent when many prototypes are stored. In any event, the matrix-entropy of the associator is less than for a hetero-associator so the storage will be limited accordingly. Another problem regards classification. An auto-associator requires

[10] The evidence for this was obtained by a "cursory" investigation conducted by the author. This analysis was not included since it depended upon erroneous independence assumptions of vector dot-products and so may have been inaccurate.

the output-dimensionality to be the same as that of the input. The present investigation indicates this condition is suboptimal for classification performance.

One method for solving both problems is to use a hetero-associator (with output-dimension smaller than that of the input) but feed back the output information in some constructive fashion. However, even if this can be done, the amount of output information must be sizable in comparison with the amount of input information present at the start of the auto-association process. If the amount of output information is less than 1/2 or 1/3 of the amount of input information, the incremental increase in information available at the output after several "iterations" of the auto-associator will be only marginally better than that available to begin with. The author believes that the auto-associator will therefore have greatly improved classification performance for light storage loads but will not gain much storage capacity as a result of the auto-associative feedback.

We also mention that theorem 1, page 26, does not apply to the auto-associator since the "retrieval-address" is not independent of the datum to be retrieved since the input is generally a partial rendition of the datum to be retrieved. The theorem could be modified to take this into account, but the bound on retrievable information will be different. The auto-associator has the advantage that the input partially specifies the output, so the auto-associator needn't "work as hard" when the input specifies a substantial portion of the output. The result should be improved classification-performance over the hetero-associator even though the auto-associator has a (perhaps marginally) smaller matrix-entropy. In any event, the author believes that the methods used to evaluate classification of "single-feature" vectors might aid quantification of the performance of the auto-associator.

The other direction of research to be mentioned is the storage of prototypes whose components are zero-mean gaussians. This is a more natural mode of storage for the outer-product associator since the output vector produced is best characterized as the proper output-prototype embedded in gaussian noise. The author believes that the analysis would begin with the noisy-signal analysis of Gallager in [12, p. 32, Example 4] and proceed with evaluation of the matrix throughput.

Lastly, we mention that associators built from other storage rules such as error correction have not been treated. This may be a much more difficult problem since evaluation of the matrix-entropy could problematic. In the event that it can be determined or approximated, the theory presented here would then be applicable for performance evaluation. The result could be a theory relevant to multi-layer error-correction systems such as the Parker/Rummelhart "backpropagation" networks.

6.3. Epilog

At this point, I'd like to let *my* editorial hair down and relate a couple interesting observations. First, notice that the prototypes were treated as vectors that were to be distinguished as exemplars of distinct categories. As such, a premium was put on their dissimilarity so that the system could tell them apart. Though this may not be desirable in all associator tasks, it points up an issue regarding the "symbol" view of intelligence. If we identify the stored prototypes as "symbols" one could view symbols as a means of performing large-scale data-compression on the environment. This not only enables a system to vastly simplify its representation of the environment, but the identification of such symbols in a cognitive system could subsequently provide a parsimonious theory of cognition (Yes, I know, "traditional AI" already knows this). Not that the identification would be easy, (if symbols can be said to exist at all, they are probably too "plastic" and malleable to be static entities) but in the associator at least, the symbols are the prototype pairs. The input-prototype reflects the system's "idea" of a most typical "object type" within a large class of objects, and the output-prototype reflects the system's representation of the object. The object at this level, is known only as it belongs to a generic *class* of objects. All other information is "discarded" as irrelevant. The analysis done here showed data-compression as a consequence of the presence of symbol/prototypes. However, the relation should go the other way as well, as evidenced by studies of "compressed", "hidden-unit" representations generated within backpropagation networks. The symbol is doubtfully an explicit feature of the brain, but is probably an emergent property of data-compression.

While I'm making conjectures about how the brain works, I might as well take a stab at the amount of information it can store. The figures obtained here are doubtfully accurate for biological brains but serves as a prediction made by the following simplistic assumptions

1. The whole brain participates in storing roughly N items where N is the number of connections in the brain.

2. The connection strengths are normally distributed with variance roughly N.

3. The effect of all connections on a neuron is the linear sum of the individual effects.

How embarassing! Anyway, assuming 10,000 connections per neuron and 10^{10} to 10^{11} neurons per brain, we get 10^{14} to 10^{15} for the number of connections. The information storable is then roughly $N\log_2 N$ or 4.5×10^{15} to 5×10^{16} bits, or roughly a billion megabytes.

The only thing that will rescue this estimation is its crudeness. The noteworthy thing though is that the theory does make a prediction. It would be interesting if in the future, a better understanding of

cognition, brain-dynamics would render better assumptions than the ones given here and if so, how these assumptions affect the estimate in relation to the one I've just made. I leave it to the reader to estimate the maximum number of stimuli the brain can possibly classify. If you come up with a number (boy, would it be big!) let me know what it is over dinner and tell me what your assumptions were. Just don't publish it as a research finding (did you know that we only use 10-percent of our brains? . . .). Well, I've put in my ten-percent, thank-you!

Appendix A
Entropy of a Binomial Random Variable

In this section, we show that the entropy of a binomial r.v. is approximated by the entropy of a corresponding normal r.v. In this development, a binomial r.v. S_n is a sum of n i.i.d. **Bernoulli trials**, where a Bernoulli trial is an r.v. with outcome 0 or 1. We will only consider binomial sums of **balanced** Bernoulli-trials, that is, Bernoulli trials whose two outcomes are equiprobable. Such a binomial r.v. has variance $n/4$, and as we will show, has entropy that approaches that of a normal r.v. of the same variance. The entropy of a normal r.v. with variance $n/4$ is $(1/2)\log_2(\pi en/2)$. Therefore the following theorem will be proven in this appendix:

Theorem 1: Let S_n be the binomial r.v. associated with the sum of n i.i.d. balanced Bernoulli-trials. Then

$$\lim_{n \to \infty} \left(H(S_n) - (1/2)\log_2(\pi en/2) \right) = 0 \qquad (A.1)$$

The rate of convergence is not treated, but numerical tests have indicated it to be fairly rapid. It would be of interest to study not only the rate of convergence, but whether or not the convergence is monotone in n. That is, one would expect that

$$\left| H(S_{n+1}) - (1/2)\log_2(\pi e(n+1)/2) \right| \leq \left| H(S_n) - (1/2)\log_2(\pi en/2) \right| \qquad (A.2)$$

for all $n = 1, 2, \ldots$.

The rate and manner of convergence are not explicitly dealt with though they possibly could be inferred from the proof that follows.

A few lemmas are needed to obtain the result. Each lemma specifies that some sequence or class of sequences exists that ensure that a specific inequality be true. Constraints on the sequences sufficient for the inequality to hold are specified by each lemma. After the proof of the lemmas, the proof of the main theorem begins by showing that a sequence exists that obeys the constraints of all the lemmas simultaneously. All the respective inequalities will then hold and they can be linked together with the

"triangle-inequality" to give the result of the theorem. Arguments used in the various proofs were motivated from developments in Feller [11] and Rudin [39].

The proofs to follow generally require that, given an arbitrarily small real number $\epsilon > 0$, some positive quantity that is a function of the positive integer n will be smaller than ϵ for all sufficiently large values of n. No generality is lost by assuming that ϵ is less than 1. *This assumption will be used throughout* (except where otherwise stated). Further, to simply the arguments and notation, we consider only even values of n. The arguments for odd n would be the same but $n/2$ would have to be replaced by $(n-1)/2$. Finally, the result of each lemma will hold when ϵ is replaced by $\epsilon/4$ since ϵ is an arbitrary positive constant. This will be instrumental in the proof of the main result.

Notationally, $\phi_\sigma(x)$ is the normal probability-density function, $1/(\sqrt{2\pi}\sigma)\cdot exp(-x^2/2\sigma^2)$ for a normal r.v. X_n with a mean equal to zero and variance σ^2 where $\sigma > 0$. We will be concerned with $\sigma = \sqrt{n}/2$ and will use this value for σ throughout. The standard normal density function $1/\sqrt{2\pi}\cdot \exp(-x^2/2)$ will be denoted $\phi(x)$.

A.1. Ignoring Tails of the Normal Entropy Integral

The entropy of the a normal r.v. with variance σ^2 is given by the integral $\int_{-\infty}^{\infty} -\phi_\sigma(x)\log_2 \phi_\sigma(x)dx$. The first lemma allows approximation of the normal entropy by ignoring the "tails" of this integral. We show that for $\sigma = \sigma(n) \equiv \sqrt{n}/2$, a positive-integer sequence $\{r_n\}$, of order $O(\sqrt{n\log_2(\log_2 n)})$ exists that grows rapidly enough so that for any positive ϵ, the integral

$$\int_{-r_n}^{r_n} -\phi_\sigma(x)\log_2 \phi_\sigma(x)dx$$

is within ϵ of the true entropy for all sufficiently large n. From this it follows that if $\{s_n\}$ is a sequence whose elements exceed those of $\{r_n\}$ for all sufficiently large n then the integral

$$\int_{-s_n}^{s_n} -\phi_\sigma \log_2 \phi_\sigma dx$$

will be within ϵ of the true entropy. This property we will call **asymptotic convergence**. In particular, if $\{s_n\}$ is of higher order than $\{r_n\}$ then the just mentioned integral has this asymptotic property. Our concern is to find a lower estimate of the order of $\{r_n\}$ that is sufficient to guarantee asymptotic convergence. The following lemma and its corollary state the result.

Lemma 2: For each $n = 1, 2, \ldots,$ let X_n be a normal r.v. with variance σ^2 where $\sigma = \sqrt{n}/2$. Given $\epsilon > 0$ there exists a positive-integer sequence $\{r_n\}$ of order $O(\sqrt{n \log_2 (\log_2 n)})$ with the following properties:

1. **First property:** There exists a positive integer N_1 such that if $n \geq N_1$ then

$$| H(X_n) - \int_{-r_n}^{r_n} -\phi_\sigma(x) \log_2 \phi_\sigma(x) dx | < \epsilon \tag{A.3}$$

2. **Second property:** If $\{s_n\}$ has the property that for some positive integer N_2, $n \geq N_2$ implies $s_n \geq r_n$ then $\{s_n\}$ has the first property.

Proof: For any n the entropy of X_n is defined by

$$H(X_n) = \int_{-\infty}^{\infty} -\phi_\sigma(x) \log_2 \phi_\sigma(x) dx$$

$$= \lim_{r \to \infty} \int_{-r}^{r} -\phi_\sigma(x) \log_2 \phi_\sigma(x) dx \tag{A.4}$$

Since X_n is normal with variance σ^2 the entropy $H(X_n)$ is equal to $1/2 \log_2 2\pi\sigma^2 < \infty$ [12, p. 32]. Therefore the limit above is finite and by definition of " $\lim_{r \to \infty}$ ", a positive integer r_n exists so that $r \geq r_n$ implies equation (A.3) with r_n replaced by r. We now show that for fixed $\epsilon > 0$, a positive-integer sequence $\{r_n\}$ can be chosen as an $O(\sqrt{n \log_2 (\log_2 n)})$ function of n so that property 1 holds.

Note that $\phi_\sigma(\sigma u) = 1/\sigma \cdot \phi(u)$. Substituting the variable $u = x/\sigma$ into the integral of (A.3) and letting $b_n = r_n/\sigma$, one obtains

$$\int_{-r_n}^{r_n} -\phi_\sigma(x) \log_2 \phi_\sigma(x) dx = \sigma \int_{-b_n}^{b_n} -\phi_\sigma(\sigma u) \log_2 \phi_\sigma(\sigma u) du$$

$$= \sigma \int_{-b_n}^{b_n} -\frac{\phi(u)}{\sigma} \log_2 (\phi(u)/\sigma) du$$

$$= \int_{-b_n}^{b_n} -\phi(u) \log_2 \phi(u) du + \log_2 \sigma \int_{-b_n}^{b_n} \phi(u) du \tag{A.5}$$

We denote $\int_{-b}^{b} -\phi(u)\log_2 \phi(u)du$ by $I_1(b)$ and denote $\int_{-b}^{b} \phi(u)du$ by $I_2(b)$.

If b is allowed to approach infinity, then $I_1(b)$ converges to the entropy $(1/2)\log_2(2\pi e)$ of a standard normal r.v. We can therefore choose a constant b_0 such that $b \geq b_0$ implies that $I_1(b)$ is within $\epsilon/2$ of its limit. No harm is done if for convenience we take b_0 to be larger than 1.

Since the lemma is concerned with the dependence of b_n on n as n gets large, no generality is lost by considering only $n \geq 132$ and $\epsilon < 1/4$. For such n, let[11]

$$r_n = \lceil (\sqrt{n}/2)\{\sqrt{2\log_2(4/\epsilon)}\,(\log_2(\log_2(\sqrt{n}/2)))^{1/2} + b_0\}\rceil$$

Since $n \geq 132$ and $\epsilon < 1$ the quantities under radicals are non-negative. Also b_0 is independent of σ, so that $b_n = O(\sqrt{\log_2(\log_2 n)})$. The lemma will follow if we can show for fixed $n \geq 132$ that $b \geq b_n$ implies

$$|H(X_n) - (I_1(b) + (\log_2 \sigma)\cdot I_2(b))| < \epsilon \tag{A.6}$$

Denote $\lim_{b\to\infty} I_i(b)$ by $I_i(\infty)$, $i = 1, 2$. From the derivation above one can see that $H(X_n) = I_1(\infty) + \log_2 \sigma I_2(\infty)$ so that (A.6) is equivalent to

$$|I_1(\infty) + (\log_2 \sigma)I_2(\infty) - (I_1(b) + (\log_2 \sigma)I_2(b))| < \epsilon \tag{A.7}$$

If we show that the conditions

1. $|I_1(\infty) - I_1(b)| < \dfrac{\epsilon}{2}$

2. $|I_2(\infty) - I_2(b)| < \dfrac{\epsilon}{2|\log_2 \sigma|}$

hold for $b \geq b_n$, then the left-hand-side of (A.7) satisfies the following

$$|I_1(\infty) + \log_2 \sigma I_2(\infty) - (I_1(b) + \log_2 \sigma I_2(b))|$$

[11] The restriction, $n \geq 132$ is used to diminish the chain of inequalities on the next page concerning the parameter b. It also allows use of a sequence $\{r_n\}$ whose terms are as small as possible, though this isn't necessary to obtain a suitable sequence.

$$= \left| I_1(\infty) - I_1(b) + \log_2 \sigma (I_2(\infty) - I_2(b)) \right|$$

$$\leq \left| I_1(\infty) - I_1(b) \right| + \left| \log_2 \sigma \right| \left| I_2(\infty) - I_2(b) \right|$$

$$< \frac{\epsilon}{2} + \left| \log_2 \sigma \right| \cdot \frac{\epsilon}{2|\log_2 \sigma|} = \epsilon \qquad (A.8)$$

and the conclusion will follow. Since $b_n \geq b_0$, condition 1 is satisfied by definition of b_0. We therefore need only consider condition 2.

To show condition 2 is satisfied, we observe that if $\Phi(x)$ is the standard normal distribution function then we have [11, vol. 1, p. 176]

$$1 - \Phi(x) \leq \frac{1}{\sqrt{2\pi}\, x} \exp(-x^2/2) \qquad \text{all } x > 0 \qquad (A.9)$$

Also for $b \in \mathbf{R}$, $\Phi(-b) = 1 - \Phi(b)$ so that

$$I_2(b) = \int_{-b}^{b} \phi(u)du = \Phi(b) - \Phi(-b) = 2\Phi(b) - 1 \qquad (A.10)$$

and

$$I_2(\infty) = \lim_{b \to \infty} \int_{-b}^{b} \phi(u)du = 1 \qquad (A.11)$$

This gives

$$|I_2(\infty) - I_2(b)| = |1 - (2\Phi(b) - 1)| = 2|1 - \Phi(b)| = 2(1 - \Phi(b)) \qquad (A.12)$$

We make the observation that the equation $x + y < x \cdot y$ is satisfied for all $x \geq 4$ if $y > 4/3$. Identifying x with $\log_2(4/\epsilon)$ and y with $\log_2(\log_2 \sigma)$, we see that under the assumptions for n and ϵ that have been made on the previous page, we have $x \geq 4$ and $y > 4/3$. For $b \geq b_n$, we have the following chain of inequalities:

$$b \geq b_n$$

$$\geq \sqrt{2\log_2(4/\epsilon)}\sqrt{\log_2(\log_2 \sigma)} + b_0$$

$$> \sqrt{2\log_2(4/\epsilon) + 2\log_2(\log_2 \sigma)}$$

$$= \sqrt{2\log_2((4/\epsilon)\log_2\sigma)}$$

Therefore $-\frac{b^2}{2} < -\log_2((4/\epsilon)\log_2\sigma)$ so that $\exp(-b^2/2) < \frac{\epsilon}{4\log_2\sigma}$[12] Now $b \geq b_n > 1$ (by choice of $b_0 > 1$) and we have by (A.9)

$$2(1-\Phi(b)) \leq \frac{2}{b\sqrt{2\pi}}\exp(-b^2/2) < 2\exp(-b^2/2) < \frac{2}{4\log_2\sigma} = \frac{\epsilon}{2\log_2\sigma}$$

Using (A.12) this gives condition 2.

To finish the proof, we note that $\{r_n\}$ as defined is $O(\sqrt{n\log_2(\log_2 n)})$. We have finished showing that the first property of $\{r_n\}$ holds for $N_1 = 132$.

If $\{s_n\}$ is a sequence and N_2 a positive integer such that $n \geq N_2$ implies $s_n \geq r_n$, then set $N = \max\{N_1, N_2\}$. Since $-\phi_\sigma(x)\log_2\phi_\sigma(x) > 0$ for all x, it follows that for $n \geq N$

$$H(X_n) = \int_{-\infty}^{\infty} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx$$

$$\geq \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx$$

$$\geq \int_{-r_n}^{r_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx$$

so that

$$\left| H(X_n) - \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx \right|$$

$$\leq \left| H(X_n) - \int_{-r_n}^{r_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx \right| < \epsilon$$

From this we see that $\{s_n\}$ has property 1 mentioned in the statement of the lemma.

---

[12] We get away with freely intermixing base-2 and natural-base logarithms due to the use of the inequality. That is, for $x < 1$, we have that $y < \log_2 x$ implies $\exp(y) < \exp((\log_2 e)\ln x) = x^{\log_2 e} < x$. In this case, $y = -b^2/2$ and $x = \epsilon/(4\log_2\sigma)$.

The following result is immediate

Corollary: If $\{s_n\}$ is a sequence of order larger than $O(\sqrt{n\log_2(\log_2 n)})$, then there is a positive integer N such that $n \geq N$ implies[13]

$$| H(X_n) - \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2 \phi_\sigma(x)dx | \quad < \quad \epsilon$$

A.2. Discretization of the Normal Entropy Integral

The statement and proof of the next lemma use notation borrowed from Rudin in [39, Ch. 6] in his development of the Reimann-Stieltjes integral. The arguments he gives in theorem 6.8 [39, p. 125] for the integrability of a continuous function on a closed interval is extended to our situation. We desire to approximate an integral with a Reimann-sum, however the limits of integration are not fixed and the integrand varies with the number of points on which we sum. Our notation, which is only slightly different from Rudin's, is as follows. If $b > 0$ then a **partition** P of the closed interval $[-b, b]$ is a finite set of points $\{x_i\}_{i=-r}^{r}$ such that $-b = x_{-r} \leq x_{-r+1} \leq \ldots \leq x_r = b$. If $f(x)$ is a continuous function defined over $[-b, b]$, its maximum and minimum are attained over any closed interval in the domain of f so we put $M_{fi} = \max_{[xi,xi+1]} f(x)$, $m_{fi} = \min_{[xi,xi+1]} f(x)$, $i = -r,-r+1,\ldots,r-1$. The quantities $U_b(P,f)$, and $L_b(P,f)$ will denote the sums

$$U_b(P,f) = \sum_{i=-r}^{r-1} M_{fi}(x_{i+1} - x_i) \qquad \text{and} \qquad L_b(P,f) = \sum_{i=-r}^{r-1} m_{fi}(x_{i+1} - x_i)$$

If $\inf_P U_b(P,f)$ and $\sup_P L_b(P,f)$ are finite and have the same value, their common value is called the **Reimann-Stieltjes integral** of f over $[-b, b]$ denoted by $\int_{-b}^{b} f(x)dx$. From the definition of the integral just given, it is apparent that for any fixed P

$$L_b(P,f) \leq \int_{-b}^{b} f(x)dx \leq U_b(P,f)$$

Also the same bounds apply to the sum $\sum_{-r}^{r-1} f(x_i)(x_{i+1} - x_i)$ since $m_{fi} \leq f(x_i) \leq M_{fi}$ for $i = -r,-r+1,\ldots,r-1$.

---

[13] By "larger than $O(f(n))$" where $f(n) > 0$, we mean a sequence $\{s_n\}$ such that for any constant $C > 0$ there is an N so that $n \geq N$ implies $s_n > C \cdot f(n)$.

Before proceeding to the lemma we state the following propositions.

Proposition 3: For any $\sigma > 0$ the functions ϕ and $f(x) \equiv -\phi(x)\log_2 \phi(x)$ have bounded first-derivatives over the domain $\mathbf{R}$.

One can show that both $|f'(x)|$ and $|\phi'(x)|$ are continuous over $\mathbf{R}$ and approach zero as $x \to \pm\infty$. These together imply boundedness over $\mathbf{R}$. The second proposition is

Proposition 4: Let g be a function differentiable over a connected domain $D \subseteq \mathbf{R}$ and let B be a positive constant so that the derivative g' satisfies $|g'(x)| \leq B$ over D. Then g is uniformly continuous on D with $|g(x) - g(y)| \leq B \cdot |x - y|$ for all $x, y \in D$.

Proof: Because g is differentiable, it it continuous and so integrable over finite intervals. We have the following inequalities

$$|g(x) - g(y)| = \left| \int_y^x g(u)du \right| \leq \int_y^x |g(u)|du \leq B \cdot |x - y|$$

yielding the desired result.

We now state and prove

Lemma 5: Let $\sigma = \sqrt{n}/2$ and let $\{r_n\}$ be a sequence of positive integers such that $b(n) \equiv r_n/\sigma$ is $o(\sqrt{n}/\log_2 n)$. Given $\epsilon > 0$, there exists a positive integer N such that $n \geq N$ implies

$$\left| \int_{-r_n}^{r_n} -\phi_\sigma(x)\log_2 \phi_\sigma(x)dx - \sum_{i=-r_n}^{r_n} -\phi_\sigma(i)\log_2 \phi_\sigma(i) \right| < \epsilon \qquad (A.13)$$

Proof: We continue to use $f(x) \equiv -\phi(x)\log_2 \phi(x)$. As shown in the previous lemma, the integral in equation (A.13) is the sum of $I_1(b(n))$ and $\log_2 \sigma \cdot I_2(b(n))$ where the functions I_1 and I_2 were defined on page 86. In a similar fashion, one has

$$\sum_{i=-r_n}^{r_n} -\phi_\sigma(i)\log_2 \phi_\sigma(i) = \frac{1}{\sigma} \sum_{i=-r_n}^{r_n} f(i/\sigma) + \frac{1}{\sigma}\log_2 \sigma \sum_{i=-r_n}^{r_n} \phi(i/\sigma) \qquad (A.14)$$

Let $S_1(n)$ and $S_2(n)$ denote the first and second sums on the right hand side respectively. The lemma will follow if we can find an N so that $n \geq N$ implies

$$|I_1(b) - \frac{1}{\sigma}S_1(n)| < \frac{\epsilon}{2} \tag{A.15}$$

$$\log_2 \sigma |I_2(b) - \frac{1}{\sigma}S_2(n)| < \frac{\epsilon}{2} \tag{A.16}$$

To obtain, this we will require that N be large enough so that

$$\frac{1}{\sigma}f(r_n/\sigma) < \frac{\epsilon}{4} \tag{A.17}$$

$$\frac{1}{\sigma}(\log_2 \sigma)\phi(r_n/\sigma) < \frac{\epsilon}{4} \tag{A.18}$$

for all $n \geq N$. From proposition 3, we have the numbers $B_1 = \max_{\mathbf{R}} |f'(x)|$ and $B_2 = \max_{\mathbf{R}} |\phi'(x)|$. Let N_1, N_2 be integers such that

1. $N_1 > \dfrac{(16B_1 b(N_1))^2}{\epsilon^2}$

2. $N_2 > \dfrac{(16B_2 b(N_2))^2}{\epsilon^2}(\log_2 \sqrt{N_2}/2)^2$

and so that all $n \geq N_i$ satisfies each of these when substituted for N_i, $i = 1, 2$. We also require that N_1 is large enough that $n \geq N_1$ implies relation (A.17) and N_2 is large enough that $n \geq N_2$ implies relation (A.18). Such numbers N_1, N_2 exist since $(b(n))^2$ and $(b(n)\log_2(\sqrt{n}/2))^2$ are $o(n)$ and the left-hand-sides of (A.17), (A.18) are $o(1)$.

Fix $n \geq \max\{N_1, N_2\}$ and for notational convenience let $r \equiv r_n$ and $b \equiv b(n)$. Let $P = \{x_i\}_{-r}^{r}$ be the partition of $[-b, b]$ with $x_i = i/\sigma$, $i = -r, -(r-1), \ldots, r$ (remember $r = b\sigma$ by definition of b). Notice $x_{i+1} - x_i = 1/\sigma = 2/\sqrt{n}$. To show (A.15), we use the fact that $n \geq N_1$. Now $M_{fi} - m_{fi} = f(x) - f(y)$ for some $x, y \in [x_i, x_{i+1}]$ and we have $|x - y| \leq 2\sqrt{n}$. From this one obtains $M_{fi} - m_{fi} \leq B_1 \cdot 2/\sqrt{n}$ by proposition 4. Since $n \geq N_1$, n satisfies item 1 above so that $1/\sqrt{n} < \frac{\epsilon}{16B_1 b}$ and we can write

$$M_{fi} - m_{fi} \leq B_1 \frac{2}{\sqrt{n}} < \frac{\epsilon}{8b}$$

From this it follows that

$$U_b(P,f) - L_b(P,f) = \sum_{i=-r}^{r-1} (M_{fi} - m_{fi})(x_{i+1} - x_i)$$

$$< \frac{\epsilon}{8b} \sum_{i=-r}^{r-1} (x_{i+1} - x_i) = \frac{\epsilon}{8b} 2b = \frac{\epsilon}{4}$$

Also

$$\frac{1}{\sigma} S_1(n) = \frac{1}{\sigma} \sum_{i=-r}^{r} f(i/\sigma) = \sum_{i=-r}^{r-1} f(x_i)(x_{i+1} - x_i) + \frac{1}{\sigma} f(r/\sigma)$$

$$= Q_1(n) + \frac{1}{\sigma} f(r/\sigma)$$

where $Q_1(n)$ is the sum $\sum_{i=-r}^{r-1} f(x_i)(x_{i+1} - x_i)$. Note that $Q_1(n)$ is bounded above and below by $U_b(P,f)$ and $L_b(P,f)$ respectively (by definition of these two latter quantities). By definition of the integral, $I_1(b)$ is bounded above and below by these same quantities. It follows that $|I_1(b) - Q_1(n)| < \epsilon/4$. From this and relation (A.17), we have that $(1/\sigma)S_1(n)$ is within $\epsilon/2$ of $I_1(b)$ so that (A.15) holds.

The arguement that equation (A.16) holds is similar. In this case, recall that $n \geq N_2$ so that item 2 holds. Using the notation for the function ϕ analogous to that we used for f, we have

$$M_{\phi i} - m_{\phi i} \leq \frac{2B_2}{\sqrt{n}} < \frac{\epsilon}{8b\log_2(\sqrt{n}/2)}$$

and

$$U_b(P,\phi) - L_b(P,\phi) = \sum_{i=-r}^{r-1} (M_{\phi i} - m_{\phi i})(x_{i+1} - x_i)$$

$$< \frac{\epsilon}{8b\log_2(\sqrt{n}/2)} 2b = \frac{\epsilon}{4\log_2(\sqrt{n}/2)}$$

Finally, let $Q_2(n) = \sum_{-r}^{r-1} \phi(x_i)(x_{i+1} - x_i)$ and notice that $Q_2(n)$ is bounded above and below by $U_b(P,\phi)$ and $L_b(P,\phi)$ respectively as is $I_2(b)$. Therefore we have that

$|I_2(b) - Q_2(n)| < \epsilon \cdot 4\log_2 \sigma$. The identity $(1/\sigma)S_2(n) = Q_2(n) + (1/\sigma)\cdot\phi(r/\sigma)$ and relation (A.18) then imply the inequality (A.16). The lemma follows with $N = \max\{N_1, N_2\}$.

A.3. Approximation of Binomial Entropy

A.3.1. Error Bounds for Logarithm Terms

Feller's development [11, vol. 1, p. 179-182] is expanded here for the sake of providing approximations to terms of the binomial probability function and bounds on the error of approximation. First a few observations with respect to logarithm approximation. We start with the Taylor series for $\ln(1+t)$ which is known to be

$$\ln(1+t) = t \cdot \sum_{i=0}^{\infty} \frac{(-t)^i}{i+1} \qquad 0 < |t| < 1 \tag{A.19}$$

and for $\ln(1-t)$ it is

$$-\ln(1-t) = t \cdot \sum_{i=0}^{\infty} \frac{t^i}{i+1} \qquad 0 < |t| < 1 \tag{A.20}$$

so that

$$\ln\frac{1+t}{1-t} = \ln(1+t) - \ln(1-t) = 2t \cdot \sum_{i=0}^{\infty} \frac{t^{2i}}{2i+1} \qquad 0 < |t| < 1 \tag{A.21}$$

is obtained by adding the two series in (A.19) and (A.20). See [11, vol. 1, p. 51] for details of the derivation. Subtracting $2t$ from both sides of (A.21) gives

$$\ln\frac{1+t}{1-t} - 2t = 2t^3 \cdot \sum_{i=0}^{\infty} \frac{t^{2i}}{2i+3} \qquad 0 < |t| < 1 \tag{A.22}$$

We are interested only in values of t between 0 and 1/3 so that the series in (A.22) is positive. In other words $\ln\frac{1+t}{1-t} - 2t$ is positive. Comparing this with a geometric series with $t = 1/3$, we have the chain of inequalities

$$2t^3 \sum_{i=0}^{\infty} \frac{t^{2i}}{2i+3} < \frac{2t^3}{3} \cdot \sum_{i=0}^{\infty} t^{2i} \le \frac{2t^3}{3} \sum_{i=0}^{\infty} (1/3)^{2i} = \frac{2t^3}{3} \cdot \frac{1}{1-1/9} = \frac{3t^3}{4}$$

Since the series in (A.22) contains only positive terms and the first term is 1, we also have

$\ln \frac{1+t}{1-t} - 2t > \frac{2t^3}{3}$. Putting these inequalities together we have

$$\frac{2t^3}{3} < \ln \frac{1-t}{1-t} - 2t < \frac{3t^3}{4} \qquad \text{when } 0 < t < 1/3 \tag{A.23}$$

Similarly we can evaluate $\ln(1+t) - t$ for t in the stated range. Subtraction of t from the series (A.19) yields

$$\ln(1+t) - t = -t^2 \sum_{i=0}^{\infty} \frac{(-t)^i}{i+2}$$

The series is absolutely convergent over the range of t considered.[14] One can therefore consider the terms of the series in any order without altering the sum [39, p. 78]. We group the terms of the summation in pairs to get

$$\ln(1+t) - t = -t^2 \cdot \sum_{i=0}^{\infty} t^i \left(\frac{1}{i+2} - \frac{t}{i+3} \right)$$

Since the terms of the sum are positive, $\ln(1+t) - t$ is negative. To assess its magnitude calculate

$$|\ln(1+t) - t| = \left| -t^2 \sum_{i=0}^{\infty} \frac{(-t)^i}{i+2} \right| \leq t^2 \cdot \sum_{i=0}^{\infty} \frac{|(-t)^i|}{|i+2|}$$

$$< t^2 \cdot \sum_{i=0}^{\infty} t^i = \frac{t^2}{1-t}$$

Since $0 < t < 1/3$, we have $\frac{1}{1-t} < 3/2$ and so

$$|\ln(1+t) - t| < \frac{3t^2}{2}$$

and therefore

$$\frac{-3t^2}{2} < \ln(1+t) - t < 0 \qquad 0 < t < 1/3 \tag{A.24}$$

---

[14] A series is said to be absolutely convergent if and only if it converges when each of its terms is replaced by its absolute value.

A.3.2. Expansion of Binomial Coefficients

These observations made, one can now follow the development of [11, vol. 1, Ch. VII.2], who derives an approximation to the "central" binomial coefficients. We will take n to be even throughout and set ν to be $n/2$ to simplify notation. The case for n odd would be treated similarly with $\nu = (n-1)/2$. Let $a_k = 2^{-n}\binom{n}{\nu+k}$ be the probability that the binomial sum S_n exceeds the mean, $n/2$, by k. Since a_{-k} equals a_k, we will only consider non-negative integers k. Our goal is the analysis of the error incurred when a_k is approximated by the normal density of variance $n/2$.

It is easy enough to verify that

$$a_k = a_0 \cdot \frac{\nu(\nu-1)\ldots(\nu-(k-1))}{(\nu+1)(\nu+2)\ldots(\nu+k)} \tag{A.25}$$

There are k terms in the numerator and in the denominator so we may divide each term by ν without changing the value of the fraction

$$a_k = a_0 \cdot \prod_{j=0}^{k-1} \left(1-\frac{j}{\nu}\right) \Big/ \prod_{j=1}^{k} \left(1+\frac{j}{\nu}\right) \tag{A.26}$$

For $k < \nu/3$, and $|j| \leq k$ we use the approximation $1 + j/\nu \approx \exp(j/\nu)$ to transform the product in (A.26) into

$$a_k = a_0 \exp\left[\sum_{j=1}^{k-1} \frac{-j}{\nu} - \sum_{j=1}^{k-1} \frac{j}{\nu} - \frac{k}{\nu}\right]$$

Using the fact that $\sum_{j=1}^{k-1} j = k(k-1)/2$ one has

$$a_k \approx a_0 \exp(-k^2/\nu) \tag{A.27}$$

Using Stirling's formula to approximate factorials, the term $a_0 = 2^{-n}\binom{n}{\nu}$ is approximately $\sqrt{2/\pi n}$ and we obtain the normal-density approximation to the binomial coefficient a_k

$$a_k \approx \sqrt{2/\pi n} \cdot \exp(-k^2/\nu) \tag{A.28}$$

Notice that the right-hand-side of this equation is the normal probability-density function of an r.v. X_n with variance $\sigma^2 = n/4$ evaluated at $k\sigma$ standard deviations from the mean. Allowing ϵ_1 and ϵ_2 to

represent the errors occurring in the approximation (A.27) and in that of a_0 respectively put

$$a_k = a_0 \exp(-k^2/\nu) \exp(-\epsilon_1) \tag{A.29}$$

$$a_0 = \sqrt{2/(\pi n)} \exp(\epsilon_2) \tag{A.30}$$

so that[15]

$$a_k = \sqrt{2/\pi n} \exp(-k^2/\nu) \exp(-(\epsilon_1 - \epsilon_2)) \tag{A.31}$$

This defines ϵ_1 and ϵ_2 and the relation

$$\exp(-k^2/\nu) \exp(\epsilon_1) = \prod_{j=1}^{k-1} (1 - j/\nu) \; / \; (1 + k/\nu) \prod_{j=1}^{k-1} (1 + j/\nu) \tag{A.32}$$

is obtained from equations (A.26) and (A.29). Taking logarithms of both sides

$$-k^2/\nu - \epsilon_1 = \sum_{j=1}^{k-1} \ln \frac{1 - j/\nu}{1 + j/\nu} - \ln (1 + k/\nu)$$

Using the fact that $k^2/\nu = 2\sum_{j=1}^{k-1} j/\nu + k/\nu$ we solve for ϵ_1

$$\epsilon_1 = \sum_{j=1}^{k-1} \ln \frac{1 + j/\nu}{1 - j/\nu} - \frac{2j}{\nu} + \ln \left(1 + \frac{k}{\nu}\right) - \frac{k}{\nu} \tag{A.33}$$

A.3.3. Upper Bound on Binomial Tail Coefficients

We are ready to state and prove

Proposition 6: For integers $\nu \equiv n/2$ and k in the range $\lceil \sqrt{7n} \rceil \leq k \leq n/6$, the relation $a_k \leq a_0 \exp(-k^2/\nu)$ holds.

---

[15] Here Feller omits the leading sign in the error-exponent by setting $a_k = \sqrt{2/\pi n} \exp(-k^2/\nu) \exp(\epsilon_1 - \epsilon_2)$.

Proof: The observations made in the previous section now come into play. By hypothesis, we have that $k < n^{.6}$ so $k\nu < 1/3$. We substitute $t = k\nu$ into equation (A.22) and see that the terms of the sum in equation (A.33) are positive with the j^{th} term less than $3(j\nu)^3/4$. Since $\sum_{j=1}^{k-1} j^3 = (k(k-1))^2/4$, this sum is less than

$$\frac{3}{4}\sum_{j=1}^{k-1}(j\nu)^3 = \frac{3}{16}\cdot\frac{k^4-k^2}{\nu^3} < \frac{k^4}{4\nu^3}$$

We can get a lower bound on the term to the right of the sum in equation (A.33) by putting $t = k\nu$ into equation (A.24). The sum in (A.24) is negative and larger than $-3/2(k\nu)^2$. From equation (A.33) and these bounds, we get an upper and lower bound on ϵ_1:[16]

$$-\frac{3}{2}(k\nu)^2 < \epsilon_1 < \frac{k^4}{4\nu^3} \tag{A.34}$$

On the other hand, from equation (A.23) each term of the sum in equation (A.33) is larger than $2(j\nu)^3/3$ so that for k in the stated range the sum itself is larger than

$$\frac{2}{3}\sum_{j=1}^{k-1}(j\nu)^3 = \frac{2}{3}\cdot\frac{k^4-k^2}{4\nu^3} > \frac{k^4}{8\nu^3}$$

Therefore a tighter lower bound on ϵ_1 is

$$\epsilon_1 > \frac{k^4}{8\nu^3} - \frac{3}{2}\cdot(k\nu)^2 \tag{A.35}$$

For ϵ_2, Feller [11, vol. 1, p. 182] shows that

$$\frac{1}{4n} - \frac{1}{20n^3} < \epsilon_2 < \frac{1}{4n} + \frac{1}{360n^3} \tag{A.36}$$

so that $0 < \epsilon_2 < n/3$ in any event. Combining this with the lower bound for ϵ_1 we get

$$\epsilon_1 - \epsilon_2 > \frac{k^4}{8\nu^3} - \frac{3}{2}\cdot\frac{k^2}{\nu^2} - \frac{1}{3n}$$

We set

$$\frac{k^4}{8\nu^3} - \frac{3}{2}\cdot\frac{k^2}{\nu^2} - \frac{1}{3n} > 0$$

---

[16] In this section, only the lower bound will be useful. The upper bound will be useful in a later section.

to get a sufficient condition for $\epsilon_1 - \epsilon_2$ to be positive. This condition is met for all $k \geq \sqrt{7n}$. Therefore for k in the range stated in the hypothesis, we have that the term $\exp(-(\epsilon_1 - \epsilon_2))$ of equation (A.31) is less than 1. Equation (A.31) then implies that $a_k < a_0 \exp(-k^2/n)$ and the lemma is proved.

A.4. Ignoring tails of the Binomial Entropy Sum

In this section, we state and prove a lemma (called in this section, the *tails lemma*) that shows one can approximate the binomial-entropy by summing relatively few terms of the entropy-sum. The approximation approaches the entropy of S_n as the total number n of terms gets large.

A.4.1. Relations Used in the Proof of the Tails Lemma

Before proving the last two lemmas, a few observations necessary. These relate to the error-magnitude to be encountered in the tails lemma.

Proposition 7: For t in the range $-1/3 < t < 1/3$, the relation

$$|1 - \exp(-t)| < 3/2 \cdot |t| \tag{A.37}$$

Proof: This is easily seen from the inequalities obtained from the Taylor series for $\exp(-t)$

$$|1 - \exp(-t)| = |t \cdot \sum_{i=0}^{\infty} \frac{(-t)^i}{(i+1)!}| \leq |t| \cdot |\sum_{i=0}^{\infty} \frac{(-t)^i}{(i+1)!}| \leq |t| \sum_{i=0}^{\infty} |t|^i$$

$$= |\frac{t}{1-t}| \leq \frac{1}{1-1/3} \cdot |t| = \frac{3}{2} \cdot |t|$$

One more observation must be made before pr ceeding to the lemma. Since $\lim_{x \to 0} x\log_2 x = 0$ the function $x\log_2 x$ is continuous over the closed interval $[0, 1]$ provided we define $0\log_2 0 \equiv 0$ to be consistent with the mentioned limit. Taking derivatives, ($\log_2 x = (\ln x)\log_2 e$) one can verify that the function $-x\log_2 x$ is unimodal with maximum value $e^{-1}\log_2 e$ achieved at $x = e^{-1}$. The function is continuous on the closed interval $0,1$ and so is uniformly continuous in this range.

Given $\epsilon > 0$, we seek conditions on x positive such that $|x\log_2 x| < \epsilon$.

Proposition 8: Let $\epsilon > 0$ be given. Then if $x \in [0, 1]$ and α is any number in the range $0 < \alpha < 1$ the inequality

$$x < (\alpha\epsilon e \log_2 e)^{1/(1-\alpha)} \tag{A.38}$$

implies that

$$|x\log_2 x| < \epsilon \tag{A.39}$$

Proof: Given the hypothesis, (A.38), solve for ϵ to get

$$\epsilon > 1/\alpha \cdot x^{1-\alpha} \cdot e^{-1} \cdot \log_2 e \tag{A.40}$$

Since $x^\alpha \in [0.1]$, it follows that $x^\alpha \log_2 x^\alpha \leq e^{-1}\log_2 e$. From this we have

$$|x\log_2 x| = -x\log_2 x = -x^{1-\alpha}x^\alpha \log_2 [(x^\alpha)^{1/\alpha}]$$

$$= -\frac{1}{\alpha} \cdot x^{1-\alpha}(-x^\alpha \log_2 x^\alpha) \leq \frac{1}{\alpha} \cdot x^{1-\alpha} \cdot e^{-1} \cdot \log_2 e$$

The last expression is less than ϵ by relation (A.40) so that the proof is complete.

For our purposes $\alpha = 1/2$ can be chosen to give

$$x < (e\epsilon/2\log_2 e)^2 \Rightarrow |x\log_2 x| < \epsilon \tag{A.41}$$

A.4.2. Proof of the Binomial Tails Lemma

We are now ready to state and prove the tails lemma.

Lemma 9: Given $\epsilon > 0$, there is a sequence $\{r_n\}$ of order $O(\sqrt{n\log_2 n})$ such that

$$|H(S_n) - \sum_{k=-r_n}^{r_n} -a_k \log_2 a_k| < \epsilon \tag{A.42}$$

Proof: For $n < 10$, we can take $r_n = n$. For $n \geq 10$ choose $r_n = \lfloor \sqrt{2n\log_2 n} \rfloor$ and notice $r_n + 1 > \sqrt{7n}$. Since r_n is $O(\sqrt{n\log_2 n})$, we can choose an N large enough that the following conditions hold for all $n \geq N$:[17]

1. $r_n < n/6 - 1$
2. $n \geq (2\log_2 e/(e\epsilon))$

For fixed $n \geq N$, let $k = r_n + 1$ and write the following inequalities

$$k \geq \sqrt{2n\log_2 n} = \sqrt{n\log_2 n + n\log_2 n} \geq \sqrt{n\log_2 n + n\log_2 (2\log_2 e/(e\epsilon))}$$

so that

$$k^2 \geq n\log_2 [2n\log_2 e/(e\epsilon)]$$

and

$$-2k^2/n \leq 2\log_2 (e\epsilon/(2n\log_2 e))$$

This implies

$$\exp(-2k^2/n) < (e\epsilon/(2n\log_2 e))^2$$

Since $\sqrt{7n} \leq k \leq n/6$, proposition 8 implies $a_k \leq a_0 \exp(-2k^2/n)$. Together with the fact that $a_0 < 1$ this implies for $l \geq k$:

$$a_l \leq a_k \leq a_0 \exp(-2k^2/n) \leq \exp(-2k^2/n) < (e\epsilon/(2n\log_2 e))^2$$

We see that a_l satisfies the hypothesis of proposition 8 with ϵ replaced by ϵ/n and therefore

$$|a_l \log_2 a_l| < \frac{\epsilon}{n} \qquad r_n + 1 \leq l \leq n$$

---

[17] Notice that the second condition stipulates that the left-hand-side of (A.42) will be less than any $\epsilon \geq 2\log_2 e/(en)$. Therefore, $2\log_2 e/(en)$ is roughly the maximum entropy lost when S_n is "approximated" by a random variable $S_n' = \min\{S_n, r_n\}$. We say "roughly" because we have not accounted for the fact that S_n' will equal $\pm r_n$ with a slightly higher probability than the probability that S_n will assume these two values.

is the desired upper bound on "tail" terms of the binomial-entropy sum. We can now verify the conclusion (remember, n is even)

$$|H(S_n) - \sum_{k=-r_n}^{r_n} -a_n \log_2 a_n| = |\sum_{k=-n/2}^{n/2} -a_k \log_2 a_n - \sum_{k=-r_n}^{r_n} -a_k \log_2 a_k|$$

$$= |2 \sum_{k=r_n+1}^{n/2} -a_k \log_2 a_k|$$

$$\leq 2 \sum_{k=r_n+1}^{n/2} |a_k \log_2 a_k| < n \cdot \epsilon/n = \epsilon$$

The lemma is proved. We also have the following corollary for sequences of higher order than the sequence $\{r_n\}$:

Corollary: For ϵ, $\{r_n\}$ as in the lemma, let $\{s_n\}$ be a positive integer sequence such that $n \geq s_n \geq r_n$ for all n, then

$$|H(S_n) - \sum_{k=-s_n}^{s_n} -a_k \log_2 a_k| < \epsilon$$

Proof: The terms in the sum above are all positive. Since $n \geq s_n \geq r_n$, we have

$$H(S_n) \equiv \sum_{k=-n}^{n} -a_k \log_2 a_k \geq \sum_{k=-s_n}^{s_n} -a_k \log_2 a_k$$

$$\geq \sum_{k=-r_n}^{r_n} -a_k \log_2 a_k$$

Because the leftmost quantity in this string of inequalities is within ϵ of the rightmost quantity, the result of the corollary follows.

A.5. Similarity of Binomial and Normal Entropy Approximations

We have "chopped" the tails of the normal entropy integral and then discretized it to obtain a sum as a close approximation. The tails of the binomial entropy sum were also "chopped" to obtain an approximation that is a sum of far fewer terms. We now need to show that the resulting approximations for the normal entropy and for the binomial entropy are good approximations of each other.

Lemma 10: For $n = 1, 2, \ldots$ let $\sigma = \sqrt{n}/2$ and let $\{r_n\}$ be a positive-integer sequence in $O(\sqrt{n \log_2 n})$. Given $\epsilon > 0$, there exists a positive integer N such that $n \geq N$ implies

$$\left| \sum_{k=-r_n}^{r_n} -a_k \log_2 a_k - \sum_{k=-r_n}^{r_n} -\phi_\sigma(k) \log_2 \phi_\sigma(k) \right| < \epsilon \tag{A.43}$$

Proof: The sequence $\{r_n\}$ is in $O(\sqrt{n \log_2 n})$ so we consider the case that $r_n \geq \sqrt{3n}$ for all sufficiently large n.[18] Also there exists a $C > 0$ such so that $r_n < C \cdot \sqrt{n \log_2 n}$ for all n. It follows that a positive integer N_0 can be chosen so that $\sqrt{3n} < r_n < n/6$ for all $n \geq N_0$. Let n be in this range and put $t = \epsilon_1 - \epsilon_2$ where ϵ_1, ϵ_2 are defined by equations (A.29) and (A.30) as functions of the positive integer n and $k = 1, 2, \ldots, n$. From these two equations we have that $a_k = \phi_\sigma(k) \exp(-t)$ and for $k = 1, 2, \ldots, r_n$ we can bound the terms of the difference (A.43):

$$\begin{aligned}
&\left| -a_k \log_2 a_k - (-\phi_\sigma(k) \log_2 \phi_\sigma(k)) \right| \\
&\quad = \left| -\phi_\sigma(k) \exp(-t) \log_2 (\phi_\sigma(k) \exp(-t)) - (-\phi_\sigma(k) \log_2 \phi_\sigma(k)) \right| \\
&\quad = \left| \phi_\sigma(k)(1 - \exp(-t)) \log_2 \phi_\sigma(k) + \phi_\sigma(k) \cdot t \cdot \exp(-t) \cdot \log_2 e \right| \\
&\quad \leq \left| \phi_\sigma(k) \log_2 \phi_\sigma(k) \right| \cdot \left| 1 - \exp(-t) \right| + \left| \phi_\sigma(k) \right| \cdot |t| \cdot \left| \exp(-t) \right| \cdot \left| \log_2 e \right|
\end{aligned} \tag{A.44}$$

We need upper bounds on the terms $|t|$, and $|1 - \exp(-t)|$. To get an upper bound on $|t|$, consider the following.

Since $r_n \geq \sqrt{3n}$, we have $r_n^4 / 4\nu^3 \geq 3r_n^2 / 2\nu^2$. For any $k \leq r_n$ we get

---

[18] The case that $r_n < \sqrt{3n}$ results in a smaller number of terms being summed in relation (A.43). The upper bounds for the error derived in this section would still applicable to these terms. By summing less terms the total discrepancy between the two sums in (A.43) will be less, hence the case that $r_n < \sqrt{3n}$ is subsumed by the case that $r_n \geq \sqrt{3n}$.

$$\frac{3k^2}{2\nu^2} \leq \frac{3r_n^2}{2\nu^2} \leq \frac{r_n^4}{4\nu^3}$$

also $k^4/(4\nu^3) \leq r_n^4/(4\nu^3)$. By equation (A.34) then, we have that

$$|\epsilon_1| < \frac{r_n^4}{4\nu^3}$$

Since $|\epsilon_2| < n/3$ we also have $\epsilon_2 < r_n^4/(4\nu^3)$ and so

$$|t| = |\epsilon_1 - \epsilon_2| < |\epsilon_1| + |\epsilon_2| < r_n^4/(2\nu^3)$$

In turn, $r_n^4/(2\nu^3)$ is less than $4C^4(\log_2 n)^2/n$ where C was defined at the beginning of the proof.

To get a bound on $|1 - \exp(-t)|$ we take a positive integer N_1 so that $n \geq N$ implies that $4C^4(\log_2 n)^2/n < 1/3$. Therefore we have $|t| < 1/3$ and so $|1 - \exp(-t)| < 3/2|t|$ by proposition 7.

Finally, for $|t| < 1/3$, $\exp(-t)$ is bounded. Let K be a constant so that $\exp(-t) < K$ for $|t| < 1/3$. Continuing the chain of inequalities in (A.44), noting that $|\phi_\sigma(k)| < 1$, we have

$$|\phi_\sigma(k)\log_2 \phi_\sigma(k)| \cdot |1 - \exp(-t)| + |\phi_\sigma(k)| \cdot |t| |\exp(-t)| |\log_2 e|$$

$$\leq e^{-1}(\log_2 e)\cdot(3/2)\cdot|t| + K(\log_2 e)\cdot|t|$$

$$= ((3/2)e^{-1} + K)(\log_2 e)|t|$$

$$< ((3/2)e^{-1} + K)(\log_2 e)C^4(\log_2 n)^2/n$$

$$= A(\log_2 n)^2/n$$

where A is the positive constant $((3/2)\cdot e^{-1} + K)(\log_2 e)C^4$. To finish the lemma consider again the left-hand-side of (A.43) which is seen to satisfy

$$\left| \sum_{k=-r_n}^{r_n} -a_k \log_2 a_k \; - \sum_{k=-r_n}^{r_n} -\phi_\sigma(k) \log_2 \phi_\sigma(k) \right|$$

$$\leq \sum_{k=-r_n}^{r_n} \left| -a_k \log_2 a_k - (-\phi_\sigma(k) \log_2 \phi_\sigma(k)) \right|$$

There are $2r_n + 1$ terms in this sum, each positive and less than $A(\log_2 n)^2/n$. Since $r_n \leq C \cdot \sqrt{n \log_2 n}$, the sum is less than

$$[2C\sqrt{n \log_2 n} + 1] \cdot A(\log_2 n)^2/n$$

which is $O((\log_2 n)^{5/2}/\sqrt{n})$. It follows that there is a positive integer N_2 such that if $n \geq N_2$ then

$$[2C\sqrt{n \log_2 n} + 1] A(\log_2 n)^2/n \; < \; \epsilon$$

From these inequalities, the lemma follows with $N = \max \{ N_0, N_1, N_2 \}$.

A.6. Proof of the Main Theorem

We now restate and then prove the main theorem.

Theorem 11: Let S_n be the binomial r.v. associated with the sum of n i.i.d. balanced bernoulli trials. Then

$$\lim_{n \to \infty} (H(S_n) - (1/2)\log_2(\pi e n/2)) = 0 \tag{A.45}$$

Proof: We will show that for a given $\epsilon > 0$, there exists a positive integer N such that $n \geq N$ implies

$$| H(S_n) - (1/2)\log_2(\pi e n/2) | < \epsilon \tag{A.46}$$

Lemmas 2, 5, 9, and 10 can each be restated with "ϵ" replaced by "$\epsilon/4$" in their respective relations: (A.3); (A.13); (A.42); (A.43). These lemmas will still be true when modified in this way. Each lemma required a sequence that was constrained in some way to produce that particular lemma's result. Our plan is to exhibit a sequence $\{s_n\}$ that simultaneously satisfies the constraints of all four lemmas. The inequality mentioned in the conclusion of each lemma will then be true. The triangle inequality can then be used to show that the inequality (A.46) holds.

Let $\{s_n\}$ be the sequence

$$s_n = \begin{cases} n & n \leq 10 \\ \lfloor \sqrt{2n\log_2 n} \rfloor & \text{otherwise} \end{cases}$$

This is the sequence used in the proof of lemma 9 to render the inequality (A.42) (with "ϵ" replaced by "$\epsilon/4$"). In particular, for some $N_1 \geq 0$

$$| H(S_n) - \sum_{k=-s_n}^{s_n} -a_k \log_2 a_k | < \epsilon/4$$

for all $n \geq N_1$.

Since $\{s_n\}$ is $O(\sqrt{n\log_2 n}) > O(\sqrt{n\log_2(\log_2 n)})$ the corollary to lemma 2 implies that there exists a positive integer N_2 such that for $n \geq N_2$ we have

$$| H(X_n) - \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2 \phi_\sigma(x)dx | < \epsilon/4$$

Also $s_n/\sigma = O(\sqrt{\log_2 n})$, that is, $s_n/\sigma = o(\sqrt{n}/\log_2 n)$ and by lemma 5 there exists a positive integer N_3 so that for $n \geq N_3$ we have

$$| \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2 \phi_\sigma(x)dx - \sum_{k=-s_n}^{s_n} -\phi_\sigma(k)\log_2 \phi_\sigma(k) | < \epsilon/4$$

Finally, from lemma 10, we have that there is a positive integer N_4 so that $n \geq N_4$ implies[19]

$$| \sum_{k=-s_n}^{s_n} -a_k \log_2 a_k - \sum_{k=-s_n}^{s_n} -\phi_\sigma(k)\log_2 \phi_\sigma(k) | < \epsilon/4$$

Now let $N = \max\{N_1, N_2, N_3, N_4\}$ and consider any n with $n \geq N$. Since the entropy of a normal r.v. with variance $n/4$ is $1/2\log_2(\pi en/2)$, we can write

---

[19] The requirement that $s_n \geq \sqrt{7n}$ in lemma 9 is satisfied for $n \geq 12$. We take one of N_1, N_2, N_3, N_4 to be greater than 12 so that these requirements will be met for $n \geq \max\{N_1, N_2, N_3, N_4\}$ in what follows.

$$\left|\frac{1}{2}\log_2\frac{\pi e n}{2} - H(S_n)\right| = \left|H(X_n) - H(S_n)\right|$$

$$= \left| \int_{-\infty}^{\infty} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx - \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx + \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx \right.$$

$$- \sum_{k=-s_n}^{s_n} -\phi_\sigma(k)\log_2\phi_\sigma(k) + \sum_{k=-s_n}^{s_n} -\phi_\sigma(k)\log_2\phi_\sigma(k)$$

$$\left. - \sum_{k=-s_n}^{s_n} -a_k\log_2 a_k + \sum_{k=-s_n}^{s_n} -a_k\log_2 a_k - H(S_n) \right|$$

$$\leq \left| \int_{-\infty}^{\infty} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx - \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx \right|$$

$$+ \left| \int_{-s_n}^{s_n} -\phi_\sigma(x)\log_2\phi_\sigma(x)dx - \sum_{k=-s_n}^{s_n} -\phi_\sigma(k)\log_2\phi_\sigma(k) \right|$$

$$+ \left| \sum_{k=-s_n}^{s_n} -\phi_\sigma(k)\log_2\phi_\sigma(k) - \sum_{k=-s_n}^{s_n} -a_k\log_2 a_k \right|$$

$$+ \left| \sum_{k=-s_n}^{s_n} -a_k\log_2 a_k - H(S_n) \right|$$

Since each of the four absolute-value terms is less than $\epsilon/4$ by the previous lemmas, their sum is less than ϵ. The theorem is proved.

Appendix B
Mutual Information and Vector Geometry

In this appendix, we derive a relation between the mutual information shared by two ± 1-vectors, $\mathbf{A}$ and $\mathbf{B}$, and their Hamming-distance. The vector $\mathbf{A}$ will be a balanced-Bernoulli vector and the vector $\mathbf{B}$ will be chosen at random from within a neighborhood of $\mathbf{A}$ of a given radius ρ. Vector $\mathbf{B}$ will therefore provide information about $\mathbf{A}$. We will determine the relation between the information $\mathbf{B}$ provides and the neighborhood radius.

B.1. Relation of Neighborhood-Size to Neighborhood-Radius

Let $\mathcal{A}$ be the set of n-dimensional ± 1-vectors, and for the moment, let $\mathbf{A}$ and $\mathbf{B}$ be chosen randomly from $\mathcal{A}$. We wish to know the fraction of $\mathcal{A}$ lying within a given radius ρ of $\mathbf{A}$. Toward this end, consider the ball $B(\rho)$ of vectors of $\mathcal{A}$ that are within a radius ρ of $\mathbf{A}$. Since all vectors of $\mathcal{A}$ are equiprobable outcomes of $\mathbf{B}$, we can determine the fraction of vectors lying in $B(\rho)$ by determining the probability that $\mathbf{B}$ will come from $B(\rho)$. Because these vectors are chosen at random from $\mathcal{A}$, they are balanced-Bernoulli vectors. Let X be the number of components of $\mathbf{B}$ that disagree with their counterparts in $\mathbf{A}$. The r.v. X is the Hamming-distance $HD(\mathbf{A}, \mathbf{B})$ between $\mathbf{A}$ and $\mathbf{B}$. It is binomially distributed with mean $n/2$ and variance $n/4$ [26]. By the central-limit theorem, we can approximate the cumulative binomial probabilities with a normal distribution having the same mean and variance (see Lindgren [30, p. 158]).

From this we see that the probability that $\mathbf{B}$ will lie in $B(\rho)$ is $P[X \leq \rho)$ which can be determined by the normal distribution with mean $n/2$ and variance $n/4$. Half the vectors of $\mathcal{A}$ will lie within a distance of $n/2$ of $\mathbf{A}$, so so we consider the case that $\rho < n/2$ so that $B(\rho)$ comprises less than $1/2$ of $\mathcal{A}$. If we put $Z = (X - n/2)/(\sqrt{n}/2)$, then Z is a standard normal r.v. and we can write

$$P[X \leq \rho) \;=\; P[Z \leq (\rho - n/2)/(\sqrt{n}/2)) \;=\; \Phi(-z) \qquad (B.1)$$

where z is the positive number $(n/2 - \rho)/(\sqrt{n}/2)$. It is known that for z positive (say $z \geq 3$) the approximation

$$\Phi(-z) \approx \frac{\exp(-z^2/2)}{\sqrt{2\pi}z} \tag{B.2}$$

is quite accurate. [11, v. 1, p. 175]

Now suppose we want the ball $B(\rho)$ to comprise M^{-R} of $\mathcal{A}$, where $R \geq 1$. We put $P[X \leq \rho] = M^{-R}$ in (B.1) and use the approximation (B.2) to get

$$M^{-R} = \frac{\exp(-z^2/2)}{\sqrt{2\pi}z} \tag{B.3}$$

This can be rearranged to get the "z" in the exponent in terms of the other parameters

$$z = \sqrt{2R\ln M - \ln(2\pi z^2)} \tag{B.4}$$

which is a recursive expression in z. As M grows, z should grow slowly. For large M then, the "$2R\ln M$ term under the radical should dominate so that $z \approx \sqrt{2R\ln M}$. We put this value in for the "z" under the radical in (B.4) to get

$$z \approx \sqrt{2R\ln M - \ln(4\pi R\ln M)} \tag{B.5}$$

which is a good approximation to z when M is large (this can be verified by plugging the right-hand-side of (B.5) in for z in equation (B.3)). The value of ρ is ascertained from the definition of z to be

$$\rho = \frac{n}{2} - \frac{\sqrt{n}}{2}z = \frac{n}{2} - \frac{\sqrt{n}}{2}\sqrt{2R\ln M - \ln(4\pi R\ln M)} \tag{B.6}$$

So a ball encompassing roughly M^{-R} of $\mathcal{A}$ has the radius given above.

B.2. Relation of Mutual Information to Neighborhood-Radius

Now suppose **B** is chosen at random from $B(\rho)$ rather than from $\mathcal{A}$. An observer of **B** can infer that **A** lies in a radius ρ of **B**. This radius is such that a neighborhood (or ball) about **B** comprises M^{-R} of $\mathcal{A}$. Knowledge of **B** therefore constitutes an M^R-fold decrease in the possible values of **A**. Therefore the information **B** provides about **A** is $\log_2 M^R = R\log_2 M$ bits.

With regard to the n_I-dimensional input-vectors, of an associator, the vector **A** represents an input-prototype $\mathbf{F}_k$ and **B** represents the associator-input $\mathbf{F}_k'$ chosen from $B_k(\rho)$ (see the chapter on classification, page 55). The minimum value of R allowed in this case is $\pi M/(2n_O)$ where n_O is the dimension of the associator-output and M is the number of stored associations. Plugging this in for R in (B.6) gives an upper bound for ρ

$$\rho \leq \frac{n_I}{2} - \frac{\sqrt{n_I}}{2}\sqrt{\pi M \ln M/n_O - \ln(2\pi^2 M \ln M/n_O)} \qquad (B.7)$$

If we examine the n_O-dimensional output-vectors on the other hand, the vector **A** represents the output-prototype $\mathbf{G}_k$ and **B** is the associator-output $\mathbf{G}_k''$. We want a classifier sampling **B** to be able to categorize it with **A** on the basis of **B**'s distance from **A** (see figure 5-3, page 51). It is the maximal distance ρ that **B** can be from **A** that must be determined. To find this maximal distance, recall that the minimal information that **B** must provide about **A** in this case is $\log_2 M$ bits. We can substitute the value 1 for R in equation (B.6) to get an upper bound for the distance that **B** can be from **A**. The bound is

$$\rho \leq \frac{n_O}{2} - \frac{\sqrt{n_O}}{2}\sqrt{2\ln M - \ln(4\pi \ln M)} \qquad (B.8)$$

There is a problem however. In this case, each ball about an output-prototype, of the radius on the right-hand-side of (B.8), encompasses $1/M$ of the total number of possible n_O-dimensional output-vectors. This means that each prototype has a $1/M$ chance of lying in the ball about **A**. Since there are $M-1$ output-prototypes aside from **A** itself, we would expect one of them (on average) to lie in the ball about **A**. We call this a **collision**. In the case of a collision of two output-prototypes, the ball about one prototype would largely overlap with the ball about the other. Many of the vectors within ρ of one of the prototypes would not get classified with that prototype. This problem exists for all the output-prototypes. That is, each prototype will have a collision with an average of one other when ρ is given by the right-hand-side of (B.8)

To remedy the problem, we make the radius, ρ, small enough so that each ball contains only M^2 of the output-space. Now any two output-prototypes have a $1/M^2$ chance of collision with each other. Since there are roughly $M^2/2$ possible pairs of output-prototypes, less than one such pair on average will suffer from collision. If the associator produces **B** to lie within this smaller neighborhood of **A**, then **A** will be reliably classifiable. Since the ball constitutes M^{-2} of the output space, we put

$R = 2$ in (B.6) to get

$$\rho \geq \frac{n_O}{2} - \frac{\sqrt{n_O}}{2}\sqrt{4\ln M - \ln(8\pi \ln M)} \qquad (B.9)$$

This is shown as a lower bound on ρ since it is sufficient but not necessary for proper performance. In other words, some values of ρ intermediate between that of relation (B.9) and relation (B.8) should be workable. In fact, using

$$\rho = \frac{n_O}{2} - \frac{\sqrt{n_O}}{2}\sqrt{3\ln M} \qquad (B.10)$$

would result in $O(\sqrt{M})$ collisions among the M output-prototypes so that a vanishingly small fraction of the prototypes represent "degenerate" categories. We conclude then, that large systems having stored a correspondingly large number of prototypes should be able to operate nearly optimally. That is, an output-vector, $\mathbf{B}$, will be constrained to lie within ρ_M of its output-prototype $\mathbf{A}$, where ρ_M nears the upper-bound in (B.8) as M gets large. On the other hand, for smaller M we may need a redundancy at the input that is 1-1/2 to 2 times the minimal $\pi M/(2n_O)$. This assures the output information is $(3/2)\log_2 M$ to $2\log_2 M$ respectively as required by (the respective) relations (B.10) or (B.9).

References

1. Abu-Mostafa, Yaser S. Connectivity Versus Entropy. IEEE Conference on Neural Information Processing Systems - Natural and Synthetic, IEEE, November, 1987.

2. Abu-Mostafa, Yaser S. and St. Jaques, Jeannine-Marie. "Information Capacity of the Hopfield Model". *IEEE Transactions on Information Theory IT-31, No. 4* (July 1985), 461-464.

3. Amit, Daniel J., Gutfreund, Hancoch, Somplinsky, H. "Spin-Glass Models of Neural Networks". *Physical Review A 32*, 2 (August 1985), 1007-1018.

4. Amit, Daniel J., Gutfreund, Hancoch, Somplinsky, H. "Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks". *Physical Review Letters 55*, 14 (September 1985), 1530-1533.

5. Anderson, James A., Silverstein, Jack W., Ritz, Stephen A., and Jones, Randall S. "Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model". *Psycological Review 84*, 5 (1977), 413-451.

6. Anderson, James A. "Cognitive and Psychological Computation with Neural Models". *IEEE Transactions on Systems, Man, and Cybernetics SMC-13*, 5 (September/October 1983).

7. Anderson, James A., Golden, Richard M., Murphy, Gregory L.. *S.P.I.E. Institute on Hydrid and Optical Computing, Ed. 1.1 S.Z.U.* Volume :*Concepts in Distributed Systems*. S.P.I.E., Bellingham, WA, 1986.

8. Ash, Robert B.. *Inter-Science Tracts in Pure and Applied Mathematics*. Volume 19:*Information Theory*. John Wiley and Sons, New York, New York, 1965

9. Barto, A. G. "Learning by Statistical Cooperation of Self-Interested Computing Elements". *Human Neurobiology 4* (1985), 229-256.

10. Conte, Samuel D., and de Boor, Carl. *International Series in Pure and Applied Mathematics*. Volume :*Elementary Numerical Analysis, An Algorithmic Approach, 3rd Ed.* McGraw-Hill Books, 1980.

11. Feller, William. *An Introduction to Probability Theory and its Applications 3rd. Ed.* John Wiley and Sons, New York, New York, 1968.

12. Gallager, Robert G.. *Information Theory and Reliable Communication*. John Wiley and Sons, New York, New York, 1968.

13. Golden, Richard M. *Modelling Causal Schemata in Human Memory: A Connectionist Approach*. Ph.D. Th., Dept of Psychology, Brown University, Providence Rhode Island, 1986.

14. Golden, Richard M. "The "Brain-State-in-a-Box" Neural Model is a Gradient Descent Algorithm". *Journal of Mathmatical Psychology 30*, 1 (March 1986), 73-80.

15. Golden, Richard M. "A Unified Framework for Connectionist Systems". *Biological Cybernetics 1-12* (January 1988), . Recently submitted for publication, bibliographic information on this article is incomplete.

16. Greene, Peter H. "Superimposed Random Coding of Stimulus Response Connections". *Bulletin of Mathematical Biophysics 27* (Special Issue 1965), 191-201.

17. Gross, D J., Mezard, M. "The Simplest Spin Glass". *Nuclear Physics B240*, [FS12] (1984), 431-452.

18. Grossberg, Steven. *Boston Studies in the Philosophy of Science.* Volume 70:*Studies of Mind and Brain, Neural Principles of Learning, Perception, Development, Cognition and Motor Control.* D. Reidel Publishers, Boston, Mass., 1982.

19. Grossberg, Stephen. Competive Learning: From Interactive Activation to Adaptive Resonance. Article obtained in personal communication, bibliographic information not complete.

20. Harris, Dale A. Information Theory in Neuropysiology. Dept. of Physiology, Harvard Medical School. Bibliographical Information Incomplete.

21. Hinton, Geoffrey E., and Anderson, James A.. *Parallel Models of Associative Memory.* Lawrence Eribaum Associates, 365 Broadway, Hillsdale, New Jersey 07642, 1981.

22. Hinton, Geoffrey E. "Boltzmann Machines: Constraint Satisfaction Networks that Learn". *Cognitive Science 9* (1985), 147-169.

23. Hopfield, J. J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". *National Academy of Sciences, U.S.A., Biophysics 79* (April 1982), 2554-2558.

24. Hopfield, J. J. "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons". *Proceedings of National Academy of Science, U.S.A., Biophysics 81* (May 1984), 3088-3092.

25. Hopfield, J.J. and Tank, D.W. "'Neural' Computation of Decisions in Optimization Problems". *Biological Cybernetics XX* (1985).

26. Kanerva, Pentti. *Self-Propagating Search: A Unified Theory of Memory.* Ph.D. Th., Stanford University, 1983.

27. Keeler, James D. Capacity for Patterns and Sequences in Kanerva's SDM as Compared to Other Associative Memory Models. Tech. Rept. 87.29, Research Institute for Advanced Computer Science, NASA Ames Research Center, December, 1987.

28. Kohonen, Tuevo. *Springer Series in Information Sciences.* Volume 8:*Self-Organization and Associative Memory.* Springer-Verlag, New York, New York, 1984.

29. Lansner, Anders, and Ekeberg, Orjan. "Reliability and Speed of Recall in an Associative Network". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7, No. 4* (July 1985), 490-498.

30. Lindgren, Bernard W.. *Statistical Theory, 3rd Ed.* MacMillan Publishing, New York, New York, 1976.

31. Little, W. A. "The Existence of Persistent States in the Brain". *Mathematical Biosciences 19* (1974), 101-120.

32. Little, W. A., Shaw, Gordon L. "Analytic Study of the Memory Storage Capacity of a Neural Network". *Mathematical Biosciences 39* (1978), 281-290.

33. McEliece, Robert J.. *Encyclopedia of Mathematics and its Applications.* Volume 3:*The Theory of Information and Coding.* Addison-Wesley, Reading, Mass., 1977.

34. McEliece, Robert J., Posner, Edward C., Rodemich, Eugene R., Venkatesh, Santosh S. The Capacity of the Hopfield Associative Memory. California Institute of Technology, January, 1986. Submitted to IEEE Transactions on Information Theory.

35. Minsky, Marvin and Papert, Seymour. *Perceptrons, An Introduction to Computational Geometry*. M.I.T. Press, Cambridge, Mass., 1969.

36. Parker, David B. Learning Logic. Tech. Rept. TR-47, Center for Computational Research in Economics and Management Science, M.I.T., April, 1985.

37. Pearlmutter, Barak A. and Hinton Geoffrey E. G-Maximization: An Unsupervised Learning Procedure for Discovering Regularities. Neural Networks for Computing, American Institute of Physics, 1986.

38. Rosenblatt, Frank. *Principles of Neurodynamics*. Spartan Books, New York, New York, 1962.

39. Rudin, William. *International Series in Pure and Applied Mathematics*. Volume *Principles of Mathematical Analysis, 3rd Ed.* McGraw-Hill, New York, N.Y., 1976.

40. Rumelhart, David E., McClelland, James L., and the PDP Research Group. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*. M.I.T. Press, Cambridge, Mass., 1986.

41. Schneider, Walter and Mumme, Dean C. Attention, Automatic Processing and the Compiling of Knowledge: A Two-Level Architecture for Cognition. To appear in Psychology Review.

42. Sejnowski, Terrance J. and Rosenberg, Charles R. NETtalk: A Parallel Network that Learns to Read Aloud. Tech. Rept. JHU/EECS-86/01, The Johns Hopkins University Electrical Engineering and Computer Science, 1986.

43. Shaw, Gordon L., and Roney, Kathleen J. "Analytic Solution of a Neural Network Theory Based on an Ising Spin System Analogy". *Physics Letters 74A*, 1,2 (October 1979), 146-150.

44. Shiffrin, Richard M. and Schneider, Walter. "Controlled and Automatic Human Information Processing: II. Perceptual Learning, Automatic Attending, and a General Theory". *Psychological Review 84*, 2 (1977), 127-189.

45. Tanaka, F., Edwards, S. F. "Analytic Theory of the Ground State Properties of a Spin Glass: I. Ising Spin Glass". *J. Physics F: Metal Physics 10* (1980), 2769-2778.

46. Tanaka, F., Edwards, S. F. "Analytic Theory of the Ground State Properties of a Spin Glass: I. X Y Spin Glass". *J. Physics F: Metal Physics 10* (1980), 2779-2792.

47. Viterbi, A. J. "On Coded Phase-Coherent Communications". *IRE Transactions on Space Electronics and Telemetry* (March 1961), 3-14.

Using Rules and Task Division to Augment Connectionist Learning

William L. Oliver
and
Walter Schneider

University of Pittsburgh
Learning Research and Development Center
3939 O'Hara St.
Pittsburgh, PA 15260
(412) 624-7496

This paper has been submitted for publication in the *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.*

Dedication

This work is dedicated to my parents who
put up with my twelve-year college habit.

STORAGE CAPACITY OF THE LINEAR ASSOCIATOR: BEGINNINGS OF A THEORY OF COMPUTATIONAL MEMORY

BY

DEAN C. MUMME

B.S., Massachusetts Institute of Technology, 1979
M.S., Idaho State University, 1982

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1988

Urbana, Illinois

Vita

Dean C. Mumme [redacted]. He received his Bachelor of Science in Aeronautics and Astronautics in 1979 from the Massachusetts Institute of Technology. He then studied Mathematics for four years at Idaho State University, earning a Master of Science Degree in December, 1982. After studying gradute-level mathematics for an additional year, he began working for his Ph.D. at the University of Illinois, at Urbana Champaign in August 1983. During the summer of 1985, he moved with his thesis-advisor to Pittsburgh, Pennsylvania to complete his thesis research at the Learning Research and Development Center, University of Pittsburgh. He joined the University of Idaho as Assistant Professor in August 1987 where he is currently teaching and conducting research in connectionist systems.